NAME :- SHAURYA PRAKASH SHAH

ROLL:- 001811001025

ML ASSIGNMENT 2

CODE:-

IMPORTING ALL REQUIRED LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.model_selection import learning_curve
from sklearn.preprocessing import LabelBinarizer
from sklearn import datasets
from itertools import cycle
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from sklearn.metrics import roc_curve, auc
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
```

Function to plot test and training learning curves, training samples vs fit times curve and fit times vs score curve

```
def plot_learning_curve(estimator, title, X, y, axes=None, ylim=None, c
v=None,
                        n_jobs=None, train_sizes=np.linspace(.3, .7, 5)
):
  """
```

```
  Generate 3 plots: the test and training learning curve, the training
  samples vs fit times curve, the fit times vs score curve.

  Parameters
  ----------
  estimator : estimator instance
      An estimator instance implementing `fit` and `predict` methods wh
ich
      will be cloned for each validation.

  title : str
      Title for the chart.

  X : array-like of shape (n_samples, n_features)
      Training vector, where ``n_samples`` is the number of samples and
      ``n_features`` is the number of features.

  y : array-like of shape (n_samples) or (n_samples, n_features)
      Target relative to ``X`` for classification or regression;
      None for unsupervised learning.

  axes : array-like of shape (3,), default=None
      Axes to use for plotting the curves.

  ylim : tuple of shape (2,), default=None
      Defines minimum and maximum y-values plotted, e.g. (ymin, ymax).

  cv : int, cross-validation generator or an iterable, default=None
      Determines the cross-validation splitting strategy.
      Possible inputs for cv are:

        - None, to use the default 5-fold cross-validation,
        - integer, to specify the number of folds.
        - :term:`CV splitter`,
        - An iterable yielding (train, test) splits as arrays of indice
s.

      For integer/None inputs, if ``y`` is binary or multiclass,
      :class:`StratifiedKFold` used. If the estimator is not a classifi
er
      or if ``y`` is neither binary nor multiclass, :class:`KFold` is u
sed.

      Refer :ref:`User Guide <cross_validation>` for the various
      cross-validators that can be used here.

  n_jobs : int or None, default=None
      Number of jobs to run in parallel.
```

```
        ``None`` means 1 unless in a :obj:`joblib.parallel_backend` conte
xt.
        ``-1`` means using all processors. See :term:`Glossary <n_jobs>`
        for more details.

    train_sizes : array-like of shape (n_ticks,)
        Relative or absolute numbers of training examples that will be us
ed to
        generate the learning curve. If the ``dtype`` is float, it is reg
arded
        as a fraction of the maximum size of the training set (that is
        determined by the selected validation method), i.e. it has to be
within
        (0, 1]. Otherwise it is interpreted as absolute sizes of the trai
ning
        sets. Note that for classification the number of samples usually
have
        to be big enough to contain at least one sample from each class.
        (default: np.linspace(0.1, 1.0, 5))
    """
    if axes is None:
        _, axes = plt.subplots(1, 3, figsize=(20, 5))

    axes[0].set_title(title)
    if ylim is not None:
        axes[0].set_ylim(*ylim)
    axes[0].set_xlabel("Training examples")
    axes[0].set_ylabel("Score")

    train_sizes, train_scores, test_scores, fit_times, _ = \
        learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs,
                       train_sizes=train_sizes,
                       return_times=True)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    fit_times_mean = np.mean(fit_times, axis=1)
    fit_times_std = np.std(fit_times, axis=1)

    # Plot learning curve
    axes[0].grid()
    axes[0].fill_between(train_sizes, train_scores_mean - train_scores_st
d,
                         train_scores_mean + train_scores_std, alpha=0.1
,
                         color="r")
    axes[0].fill_between(train_sizes, test_scores_mean - test_scores_std,
```

```
                        test_scores_mean + test_scores_std, alpha=0.1,
                        color="g")
  axes[0].plot(train_sizes, train_scores_mean, 'o-', color="r",
               label="Training score")
  axes[0].plot(train_sizes, test_scores_mean, 'o-', color="g",
               label="Cross-validation score")
  axes[0].legend(loc="best")

  # Plot n_samples vs fit_times
  axes[1].grid()
  axes[1].plot(train_sizes, fit_times_mean, 'o-')
  axes[1].fill_between(train_sizes, fit_times_mean - fit_times_std,
                        fit_times_mean + fit_times_std, alpha=0.1)
  axes[1].set_xlabel("Training examples")
  axes[1].set_ylabel("fit_times")
  axes[1].set_title("Scalability of the model")

  # Plot fit_time vs score
  axes[2].grid()
  axes[2].plot(fit_times_mean, test_scores_mean, 'o-')
  axes[2].fill_between(fit_times_mean, test_scores_mean - test_scores_s
td,
                        test_scores_mean + test_scores_std, alpha=0.1)
  axes[2].set_xlabel("fit_times")
  axes[2].set_ylabel("Score")
  axes[2].set_title("Performance of the model")

  return plt
```

Function to display confusion matrix, classification report and ROC and AUC curves for each classifier

```
def paraMlRoutine(classifier, X, y, tSize, target_names, dataName, grid
_para):
  classifier_org = classifier

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
 tSize)
  classifier = GridSearchCV(classifier, grid_para)

  classifier.fit(X_train, y_train)
  preds = classifier.predict(X_test)

  print("----------", dataName, "---------")
```

```
  print(classifier.best_estimator_)
  cf_matrix = confusion_matrix(y_test, preds)
  sns.heatmap(cf_matrix, annot=True)
  print(classification_report(y_test, preds, target_names=target_names)
)
  plt.show()

  lb = LabelBinarizer ()
  y = lb.fit_transform (y)
  n_classes = len(lb.classes_)
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
 tSize)
  if n_classes == 2:
     y_test = np.hstack((y_test, 1 - y_test))
     y_train = np.hstack((y_train, 1 - y_train))

  # Learn to predict each class against the other
  classifier_o = OneVsRestClassifier(classifier_org.set_params(**classi
fier.best_params_))
  y_score = classifier_o.fit(X_train, y_train).predict_proba(X_test)

  fpr = dict()
  tpr = dict()
  roc_auc = dict()
  for i in range(n_classes):
      fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
      roc_auc[i] = auc(fpr[i], tpr[i])
  colors = cycle(['blue', 'red', 'green'])
  for i, color in zip(range(n_classes), colors):
      plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
               label='ROC curve of class {0} (area = {1:0.2f})'
               ''.format(i, roc_auc[i]))
  plt.plot([0, 1], [0, 1], 'k--', lw=1.5)
  plt.xlim([-0.05, 1.0])
  plt.ylim([0.0, 1.05])
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver operating characteristic for multi-class data')
  plt.legend(loc="lower right")
  plt.show()

  return classifier.best_estimator_
```

Function to display confusion matrix and classification report of ML models without hyperparamter tuning

```
def mlRoutine(classifier, X, y, tSize, target_names, dataName):
  classifier_org = classifier
  X_org = X
  y_org = y

  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
 tSize)

  classifier.fit(X_train, y_train)
  preds = classifier.predict(X_test)

  print("----------", dataName, "---------")
  print(classifier)
  cf_matrix = confusion_matrix(y_test, preds)
  sns.heatmap(cf_matrix, annot=True)
  print(classification_report(y_test, preds, target_names=target_names)
)
  plt.show()
```

Functions to train ML models with and without hyperparameter training.

```
def clRoutine(classifier, params, X, y, names, dataName):
  classifier.set_params(**params)
  train_sizes=np.linspace(.3, .7, 5)
  for train_size in train_sizes:
    pName = dataName, train_size
    mlRoutine (classifier, X, y, train_size, names, pName)

def paraClRoutine(classifier, X, y, names, dataName, grid_para):
  train_sizes=np.linspace(.3, .7, 5)
  for train_size in train_sizes:
    pName = dataName, train_size
    clf = paraMlRoutine (classifier, X, y, train_size, names, pName, gr
id_para)
    plot_learning_curve (clf, pName, X, y).show()
```

Function to define the 3 ML models namely, SVM, MLP and Random Forest

```
def flRoutine(X, y, names, dataName):
  params_dict_svc = [{"kernel":"linear"}, {"kernel":"poly"}, {"kernel":
"rbf"}, {"kernel":"sigmoid","gamma":"auto"}]
  svc = SVC(probability=True)
```

```
  for param_dict in params_dict_svc:
    sterm = dataName,"_SVC",param_dict
    clRoutine(svc, param_dict, X, y, names, sterm)
  param_grid_svc = {
      'C': [0.1, 1, 10],
      'gamma': [1, 0.01, 0.0001],
      'kernel': ['rbf', 'linear', 'poly']
      }
  sterm = dataName,"_SVC_Best"
  paraClRoutine(svc, X, y, names, sterm, param_grid_svc)

  params_dict_mlp = [{}]
  mlp = MLPClassifier()
  for param_dict in params_dict_mlp:
    sterm = dataName,"_MLP",param_dict
    clRoutine(mlp, param_dict, X, y, names, sterm)
  param_grid_mlp = {
    'momentum': [0.6, 0.9],
    'max_iter': [200, 500],
    'learning_rate': ['constant', 'adaptive']
    }
  sterm = dataName,"_MLP_Best"
  paraClRoutine(mlp, X, y, names, sterm, param_grid_mlp)

  params_dict_rfc = [{}]
  rfc = RandomForestClassifier()
  for param_dict in params_dict_rfc:
    sterm = dataName,"_RFC",param_dict
    clRoutine(rfc, param_dict, X, y, names, sterm)
  param_grid_rfc = {
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,6,8],
    'criterion' :['gini', 'entropy']
    }
  sterm = dataName,"_RFC_Best"
  paraClRoutine(rfc, X, y, names, sterm, param_grid_rfc)
```

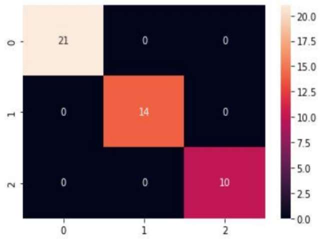
IRIS dataset :-

```
iris = datasets.load_iris ()
X = iris.data
y = iris.target
names = iris.target_names
flRoutine(X, y, names, "Iris")
```

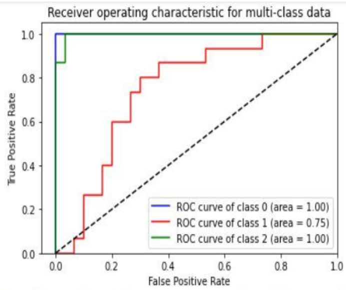
Output:-

SVM (without tuning)

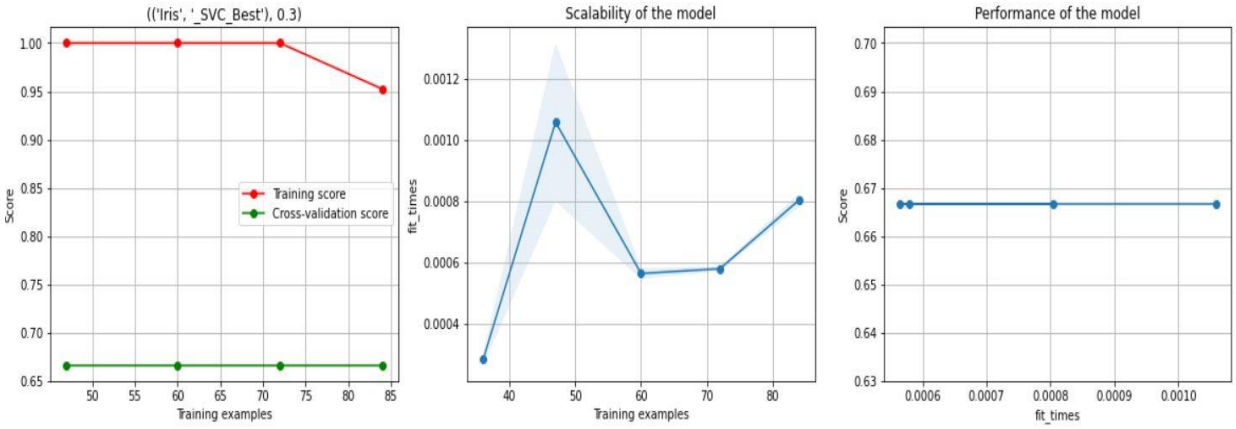
```
---------- (('Iris', '_SVC', {'kernel': 'linear'}), 0.3) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        21
  versicolor       1.00      1.00      1.00        14
   virginica       1.00      1.00      1.00        10

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```



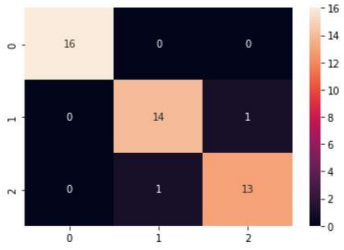
SVM (with hyperparamter tuning)




```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
```

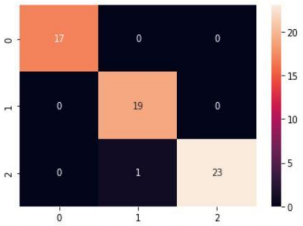



```
---------- (('Iris', '_SVC_Best'), 0.3) ---------
SVC(C=0.1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        16
  versicolor       0.93      0.93      0.93        15
   virginica       0.93      0.93      0.93        14

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45
```



MLP (without tuning)

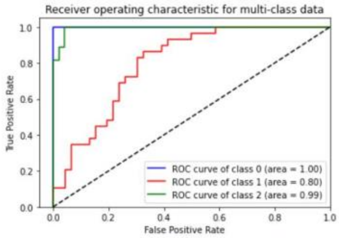
```
---------- (('Iris', '_MLP', {}), 0.39999999999999997) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        17
  versicolor       0.95      1.00      0.97        19
   virginica       1.00      0.96      0.98        24

    accuracy                           0.98        60
   macro avg       0.98      0.99      0.98        60
weighted avg       0.98      0.98      0.98        60
```



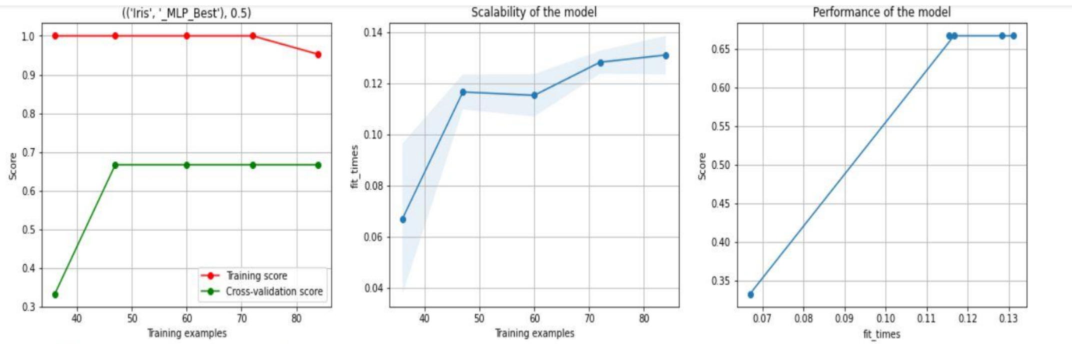
```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the
  % self.max_iter, ConvergenceWarning)
```

MLP (with hyperparamter tuning)

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
```

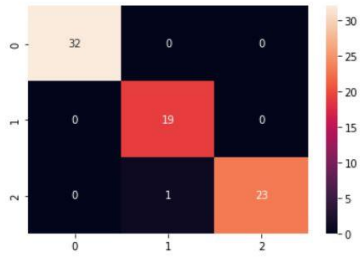



```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the opt
  % self.max_iter, ConvergenceWarning)
```



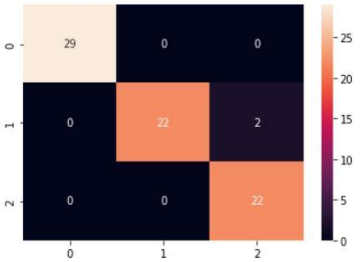

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
```

```
---------- (('Iris', '_MLP_Best'), 0.5) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        32
  versicolor       0.95      1.00      0.97        19
   virginica       1.00      0.96      0.98        24

    accuracy                           0.99        75
   macro avg       0.98      0.99      0.98        75
weighted avg       0.99      0.99      0.99        75
```

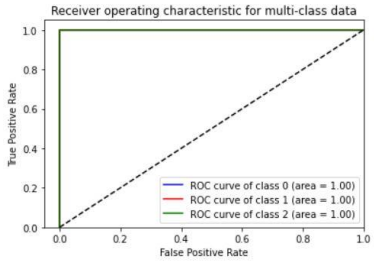


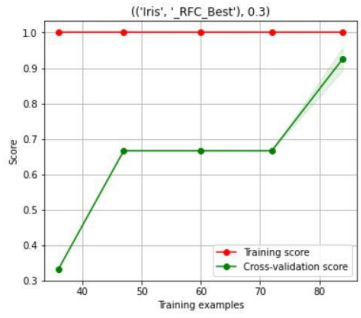
Random Forest Classifier (without tuning)

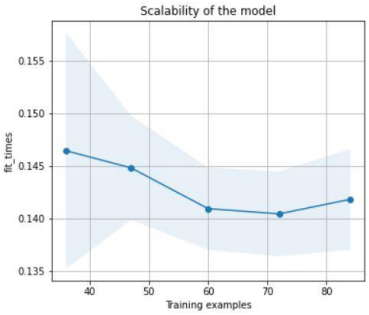
```
---------- (('Iris', '_RFC', {}), 0.5) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        29
  versicolor       1.00      0.92      0.96        24
   virginica       0.92      1.00      0.96        22

    accuracy                           0.97        75
   macro avg       0.97      0.97      0.97        75
weighted avg       0.98      0.97      0.97        75
```

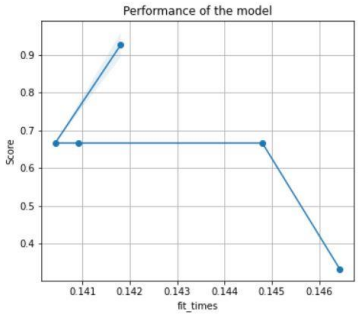


Random Forest Classifier (with hyperparamter tuning)









```
---------- (('Iris', '_RFC_Best'), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=4, max_features='log2',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        15
  versicolor       1.00      0.94      0.97        16
   virginica       0.93      1.00      0.97        14

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Receiver operating characteristic for multi-class data

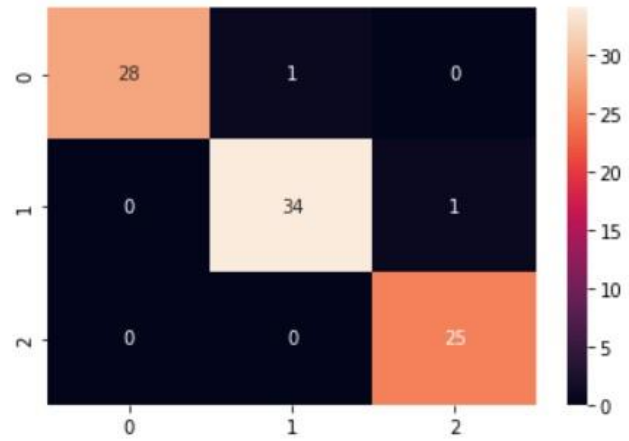
WINE dataset:-

```
wine = datasets.load_wine()
X = wine.data
y = wine.target
names = wine.target_names
flRoutine(X, y, names, "Wine")
```

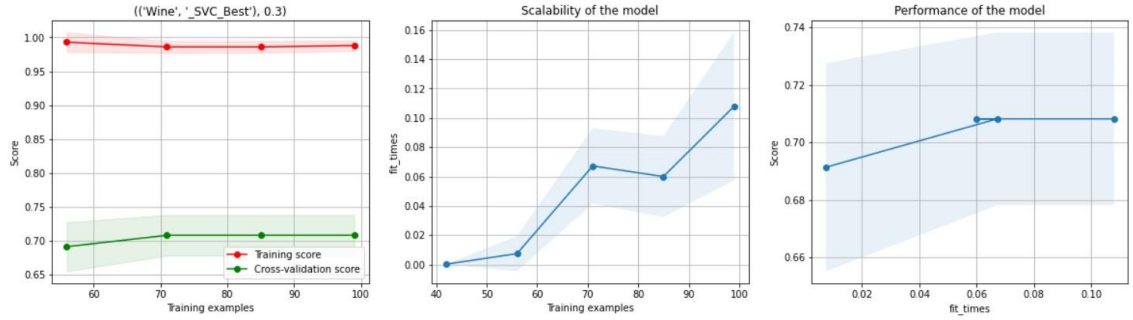
Output:-

SVM (without tuning)

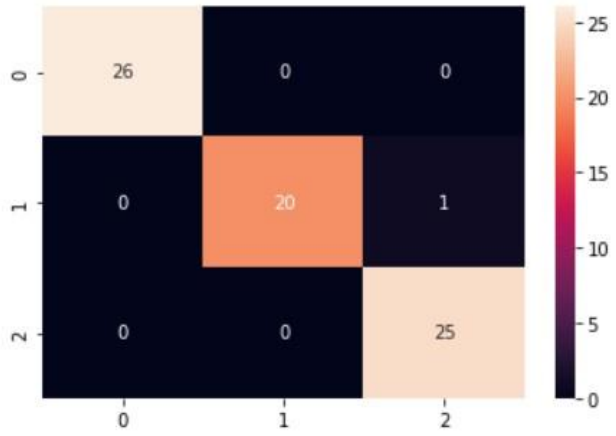
```
---------- (('Wine', '_SVC', {'kernel': 'linear'}), 0.5) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

     class_0       1.00      0.97      0.98        29
     class_1       0.97      0.97      0.97        35
     class_2       0.96      1.00      0.98        25

    accuracy                           0.98        89
   macro avg       0.98      0.98      0.98        89
weighted avg       0.98      0.98      0.98        89
```

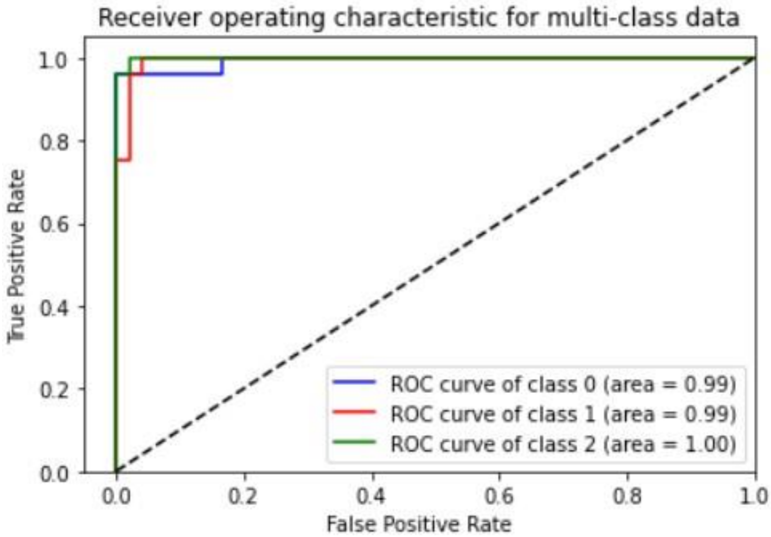


SVM (with hyperparamter tuning)



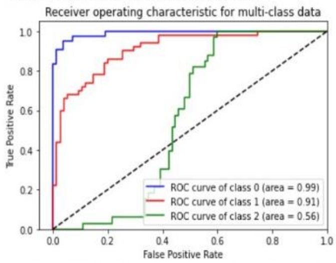

```
---------- (('Wine', '_SVC_Best'), 0.39999999999999997) ---------
SVC(C=0.1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        26
     class_1       1.00      0.95      0.98        21
     class_2       0.96      1.00      0.98        25

    accuracy                           0.99        72
   macro avg       0.99      0.98      0.99        72
weighted avg       0.99      0.99      0.99        72
```



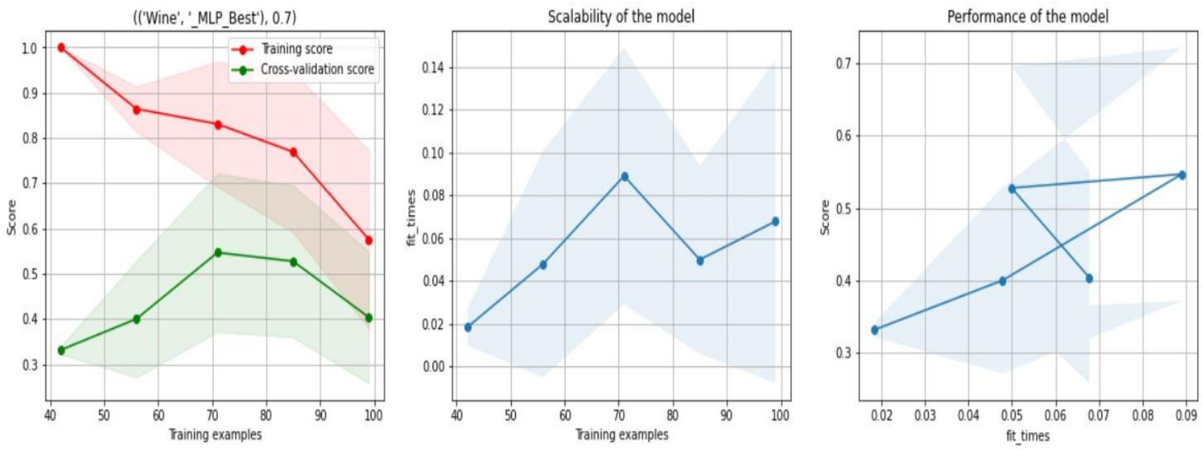


MLP (with hyperparamter tuning)

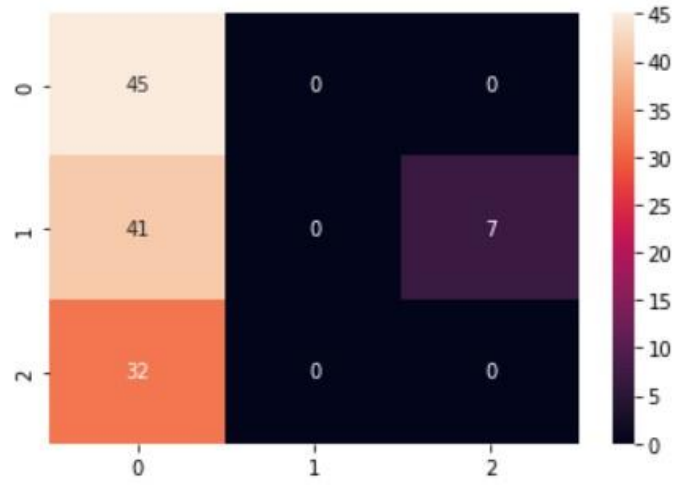
```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
```




```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optim
  % self.max_iter, ConvergenceWarning)
```

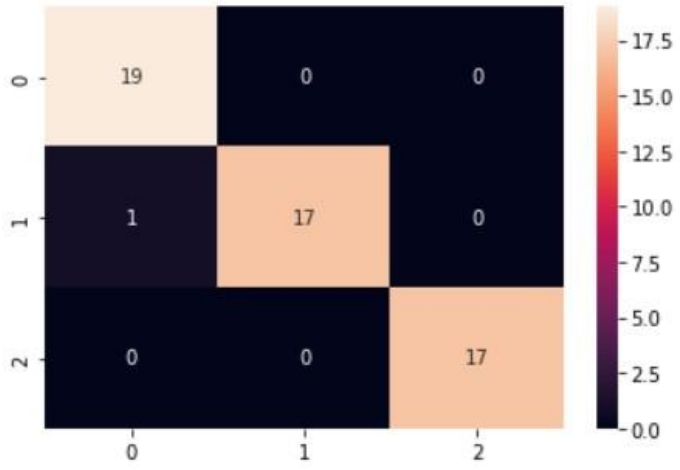


```
---------- (('Wine', '_MLP_Best'), 0.7) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

     class_0       0.38      1.00      0.55        45
     class_1       0.00      0.00      0.00        48
     class_2       0.00      0.00      0.00        32

    accuracy                           0.36       125
   macro avg       0.13      0.33      0.18       125
weighted avg       0.14      0.36      0.20       125
```

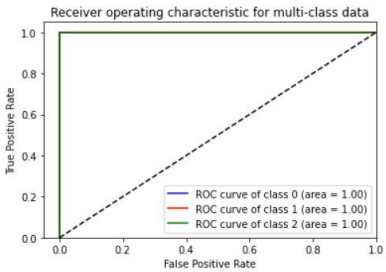


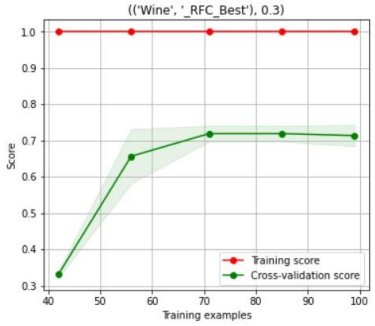
Random Forest Classifier (without tuning)

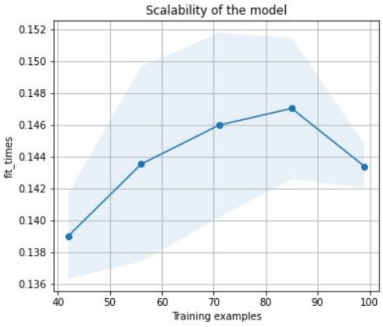
```
---------- (('Wine', '_RFC', {}), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

     class_0       0.95      1.00      0.97        19
     class_1       1.00      0.94      0.97        18
     class_2       1.00      1.00      1.00        17

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

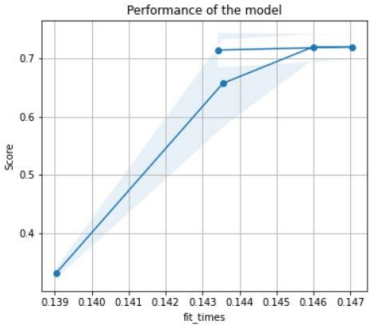


Random Forest Classifier (with tuning)

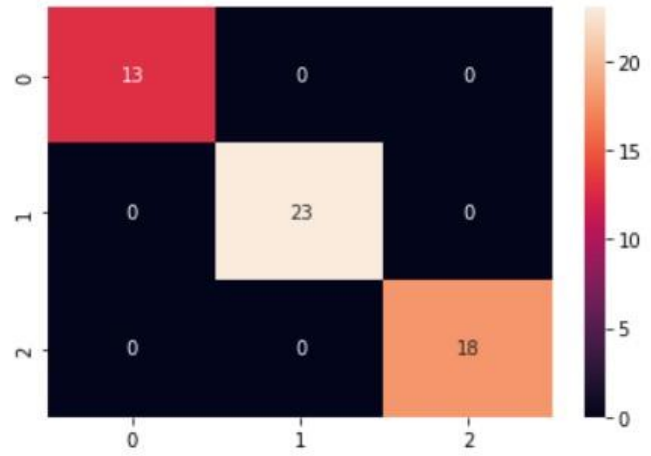








```
---------- (('Wine', '_RFC_Best'), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=6, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        13
     class_1       1.00      1.00      1.00        23
     class_2       1.00      1.00      1.00        18

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```



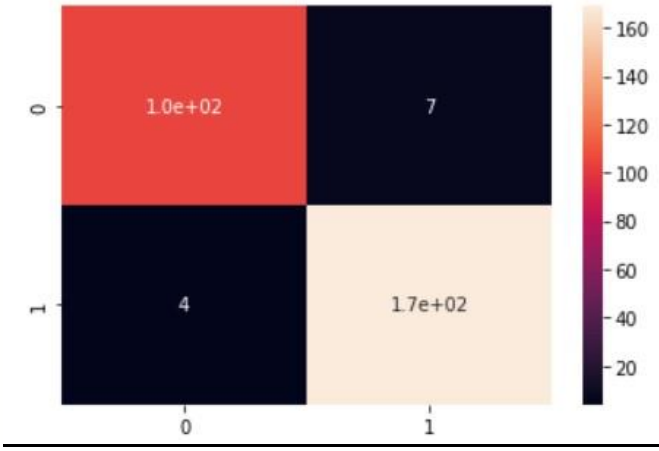
WBC dataset:-

```
wbc = datasets.load_breast_cancer()
X = wbc.data
y = wbc.target
names = wbc.target_names
flRoutine(X, y, names, "WBC")
```

Output:-

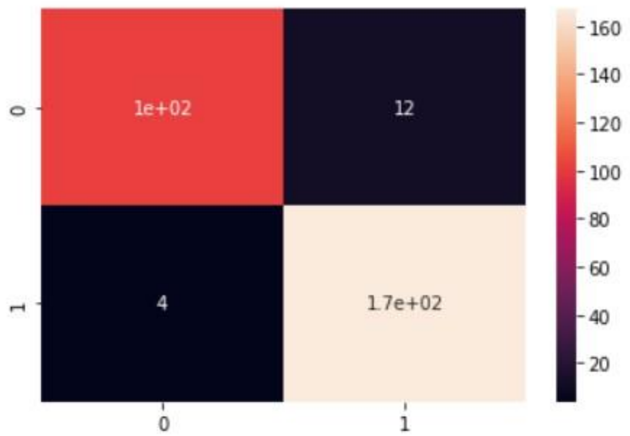
SVM (without tuning)

```
---------- (('WBC', '_SVC', {'kernel': 'linear'}), 0.5) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

   malignant       0.96      0.94      0.95       112
      benign       0.96      0.98      0.97       173

    accuracy                           0.96       285
   macro avg       0.96      0.96      0.96       285
weighted avg       0.96      0.96      0.96       285
```

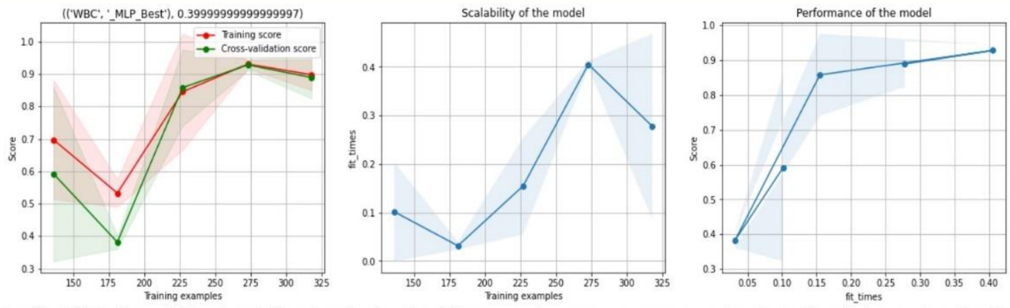


MLP (without tuning)

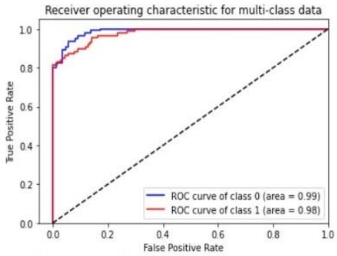
```
---------- (('WBC', '_MLP', {}), 0.5) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

   malignant       0.96      0.89      0.93       114
      benign       0.93      0.98      0.95       171

    accuracy                           0.94       285
   macro avg       0.95      0.94      0.94       285
weighted avg       0.94      0.94      0.94       285
```



MLP (without hyperparameter tuning)

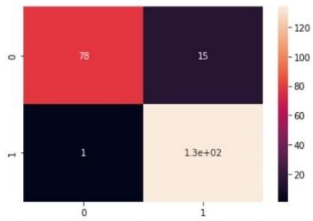



```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
```




```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
```

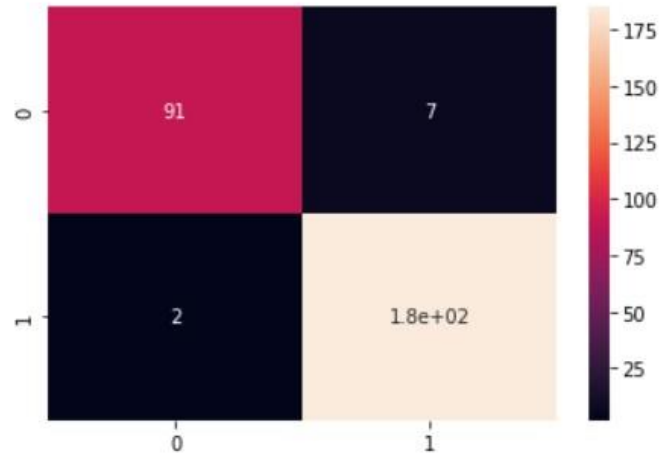
```
---------- (('WBC', '_MLP_Best'), 0.39999999999999997) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

   malignant       0.99      0.84      0.91        93
      benign       0.90      0.99      0.94       135

    accuracy                           0.93       228
   macro avg       0.94      0.92      0.93       228
weighted avg       0.94      0.93      0.93       228
```




```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
```

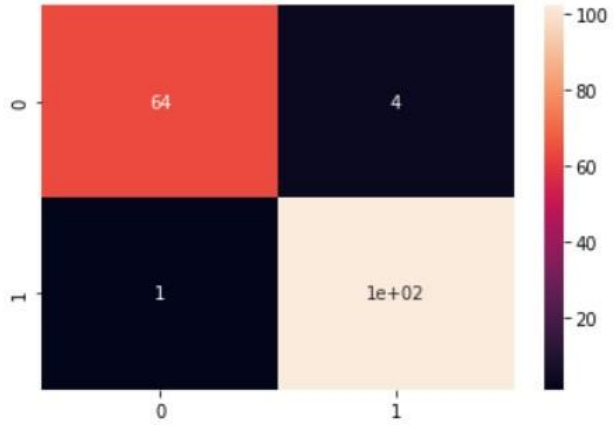
Random Forest Classifier (without tuning)

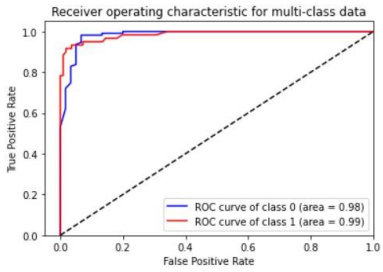
```
---------- (('WBC', '_RFC', {}), 0.5) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

   malignant       0.98      0.93      0.95        98
      benign       0.96      0.99      0.98       187

    accuracy                           0.97       285
   macro avg       0.97      0.96      0.96       285
weighted avg       0.97      0.97      0.97       285
```

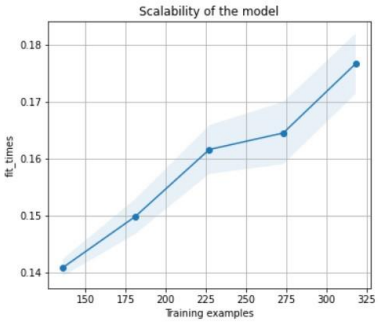


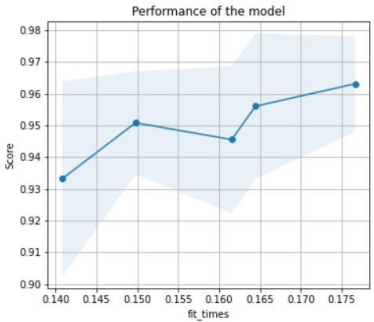
Random Forest Classifier(with hyperparameter tuning)

```
---------- (('WBC', '_RFC_Best'), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=8, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

   malignant       0.98      0.94      0.96        68
      benign       0.96      0.99      0.98       103

    accuracy                           0.97       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.97      0.97      0.97       171
```









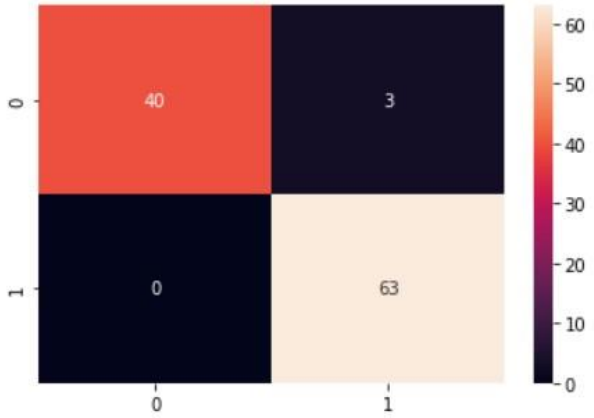
Ionosphere dataset:-

```
df=pd.read_csv('https://archive.ics.uci.edu/ml/machine-learning-databases/ionosphere/ionosphere.data',header=None)
X=df.iloc[:,0:33]
y=df.iloc[:,34]
X = X.to_numpy()
y = y.to_numpy()
names = ["g", "b"]
flRoutine(X, y, names, "Ionosphere")
```

Output:-

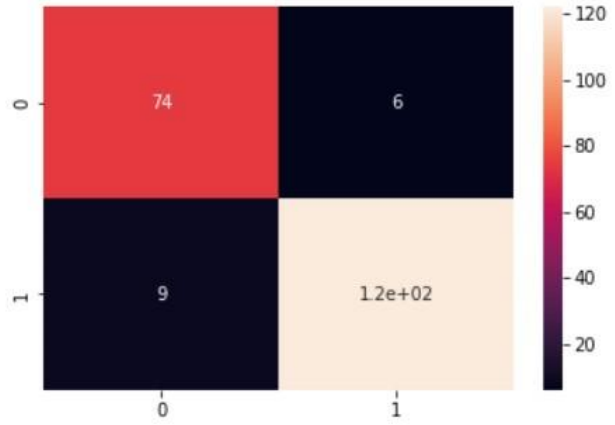
SVM (without tuning)

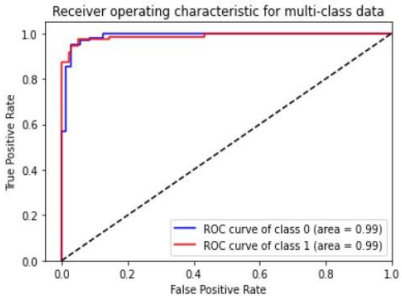
```
---------- (('Ionosphere', '_SVC', {'kernel': 'rbf'}), 0.3) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

           g       1.00      0.93      0.96        43
           b       0.95      1.00      0.98        63

    accuracy                           0.97       106
   macro avg       0.98      0.97      0.97       106
weighted avg       0.97      0.97      0.97       106
```

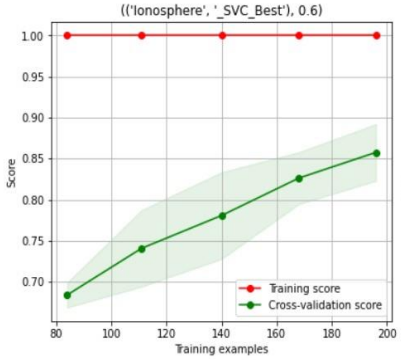


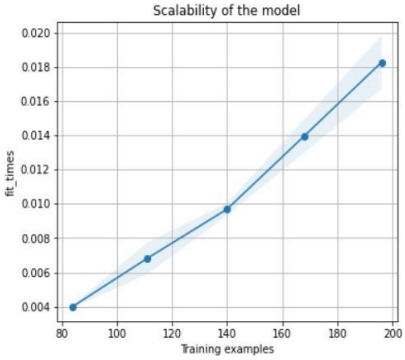
SVM (with hyperparameter tuning)

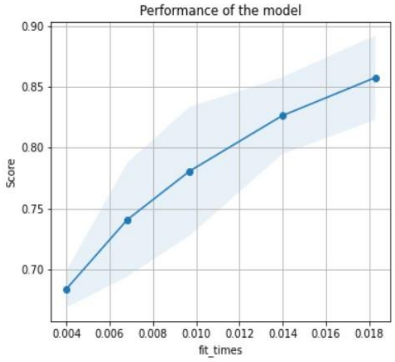
```
---------- (('Ionosphere', '_SVC_Best'), 0.6) ---------
SVC(C=10, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf', max_iter=-1,
    probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

           g       0.89      0.93      0.91        80
           b       0.95      0.93      0.94       131

    accuracy                           0.93       211
   macro avg       0.92      0.93      0.93       211
weighted avg       0.93      0.93      0.93       211
```









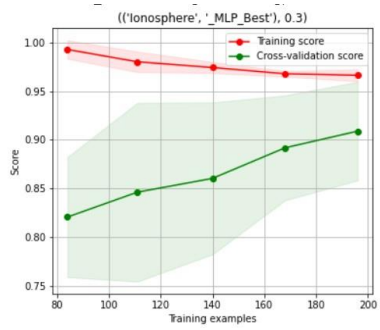


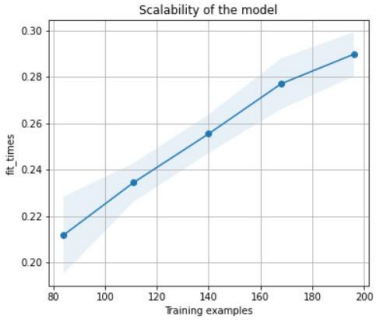
MLP (without tuning)

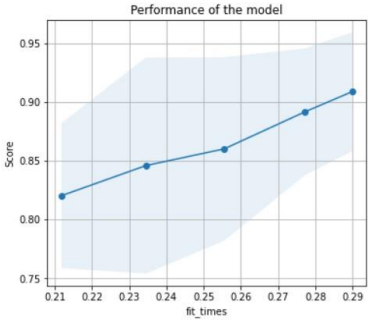
```
---------- (('Ionosphere', '_MLP', {}), 0.3) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

           g       0.94      0.82      0.88        40
           b       0.90      0.97      0.93        66

    accuracy                           0.92       106
   macro avg       0.92      0.90      0.91       106
weighted avg       0.92      0.92      0.91       106
```

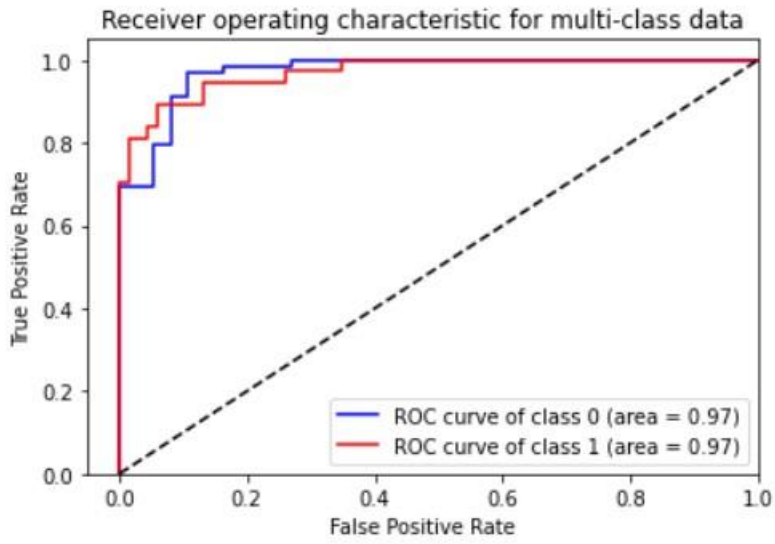
MLP (with hyperparameter tuning)

```
---------- (('Ionosphere', '_MLP_Best'), 0.3) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

           g       0.97      0.85      0.91        40
           b       0.92      0.98      0.95        66

    accuracy                           0.93       106
   macro avg       0.94      0.92      0.93       106
weighted avg       0.94      0.93      0.93       106
```



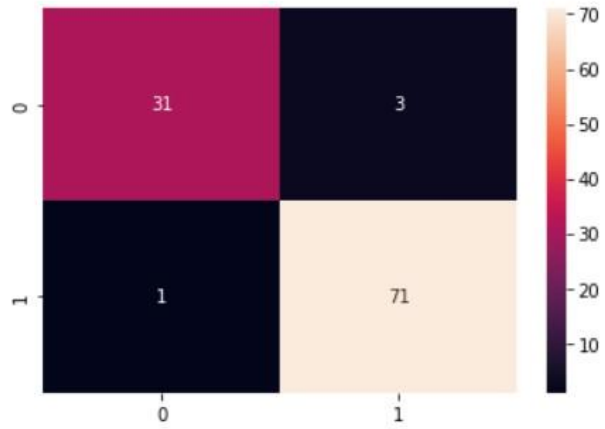






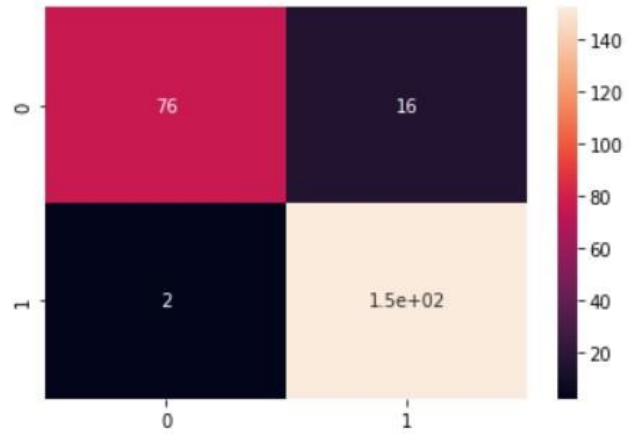
Random Forest Classifier (without tuning)

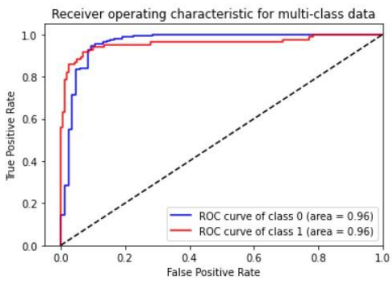
```
---------- (('Ionosphere', '_RFC', {}), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

           g       0.97      0.91      0.94        34
           b       0.96      0.99      0.97        72

    accuracy                           0.96       106
   macro avg       0.96      0.95      0.96       106
weighted avg       0.96      0.96      0.96       106
```

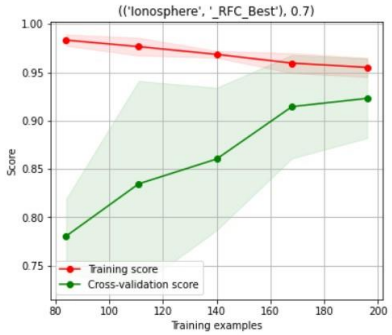


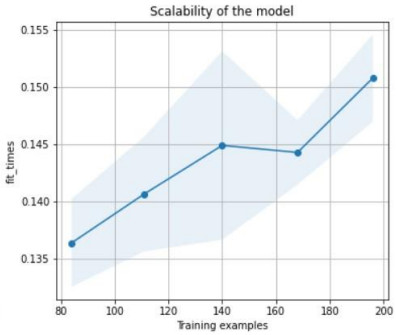
Random Forest Classifier (with hyperparamter tuning)

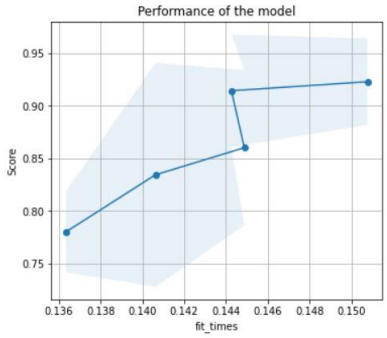
```
---------- (('Ionosphere', '_RFC_Best'), 0.7) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=4, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

           g       0.97      0.83      0.89        92
           b       0.90      0.99      0.94       154

    accuracy                           0.93       246
   macro avg       0.94      0.91      0.92       246
weighted avg       0.93      0.93      0.93       246
```











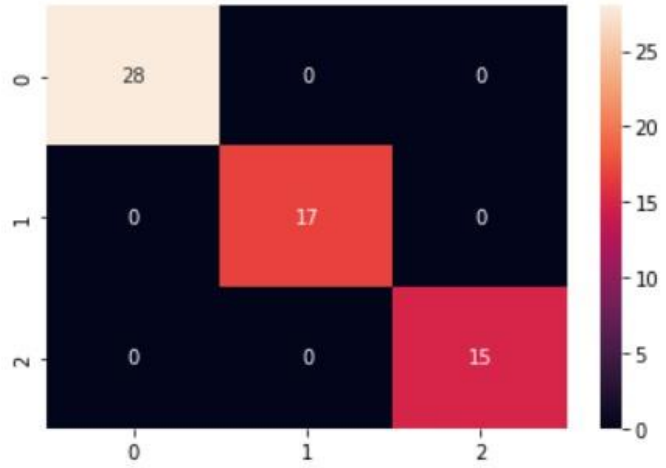
IRIS dataset with PCA:-

```
scaler = MinMaxScaler()
X=scaler.fit_transform(X)

pca = PCA()
X_new = pca.fit_transform(X)
flRoutine(X_new, y, names, "Iris_PCA")
```

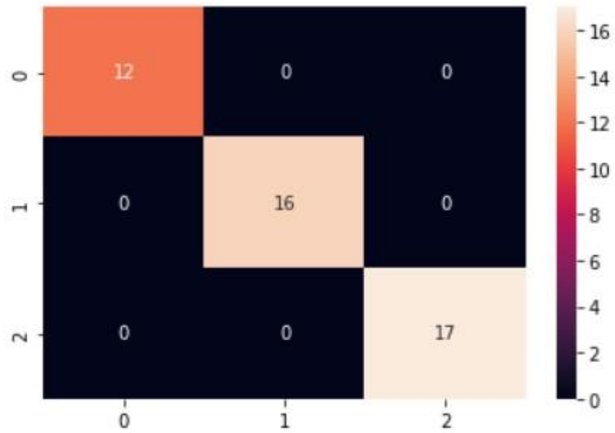
Output:-

SVM (without tuning)

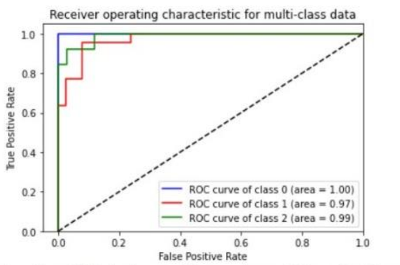
```
---------- (('Iris_PCA', '_SVC', {'kernel': 'poly'}), 0.39999999999999997) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        28
  versicolor       1.00      1.00      1.00        17
   virginica       1.00      1.00      1.00        15

    accuracy                           1.00        60
   macro avg       1.00      1.00      1.00        60
weighted avg       1.00      1.00      1.00        60
```



```
---------- (('Iris_PCA', '_SVC', {'kernel': 'rbf'}), 0.3) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        12
  versicolor       1.00      1.00      1.00        16
   virginica       1.00      1.00      1.00        17

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

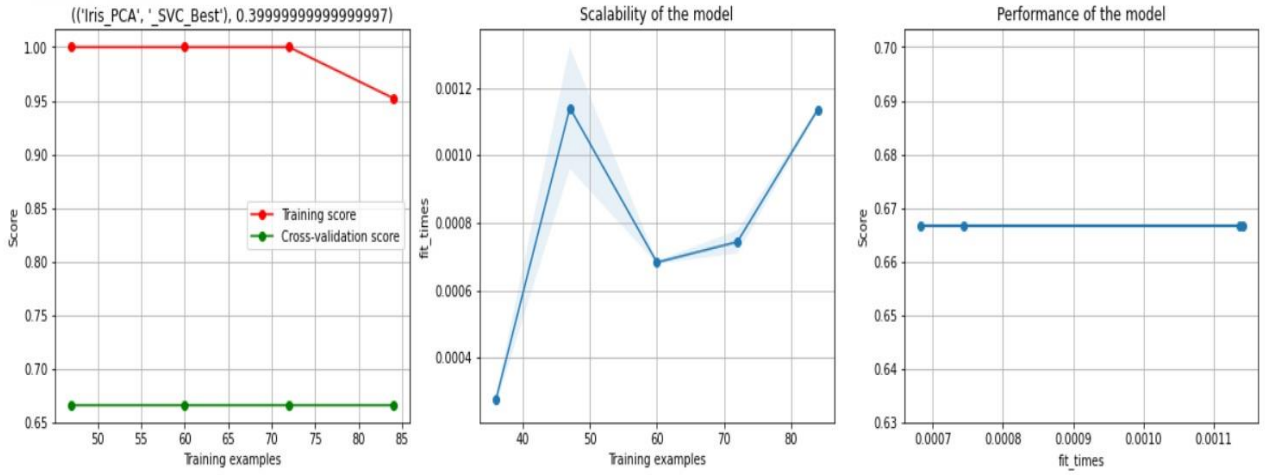


SVM (with hyperparameter tuning)

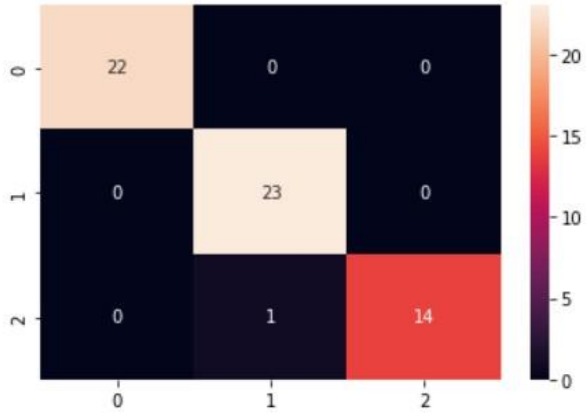



```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parameters
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
```

FitFailedWarning)

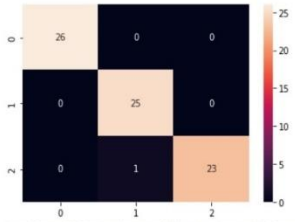



```
---------- (('Iris_PCA', '_SVC_Best'), 0.39999999999999997) ---------
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf', max_iter=-1,
    probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        22
  versicolor       0.96      1.00      0.98        23
   virginica       1.00      0.93      0.97        15

    accuracy                           0.98        60
   macro avg       0.99      0.98      0.98        60
weighted avg       0.98      0.98      0.98        60
```



MLP (without tuning)

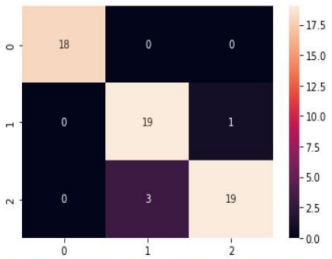
```
  % self.max_iter, ConvergenceWarning)
---------- (('Iris_PCA', '_MLP', {}), 0.5) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        26
  versicolor       0.96      1.00      0.98        25
   virginica       1.00      0.96      0.98        24

    accuracy                           0.99        75
   macro avg       0.99      0.99      0.99        75
weighted avg       0.99      0.99      0.99        75
```




```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimiz
  % self.max_iter, ConvergenceWarning)
```

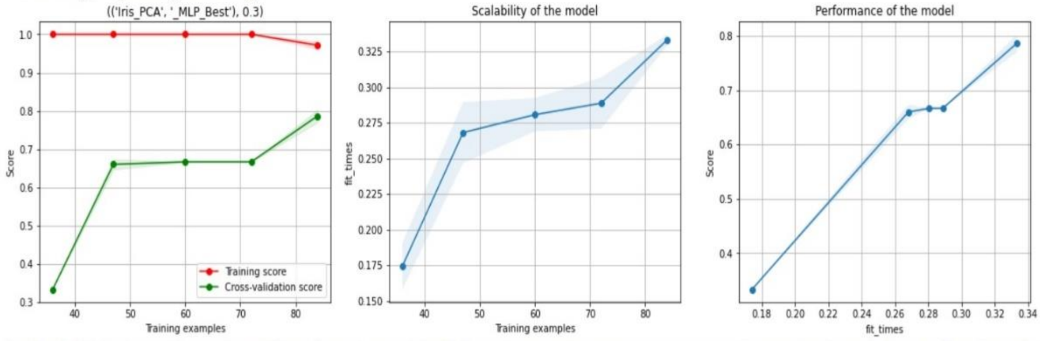
MLP (with hyperparamter tuning)

```
---------- (('Iris_PCA', '_MLP_Best'), 0.39999999999999997) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        18
  versicolor       0.86      0.95      0.90        20
   virginica       0.95      0.86      0.90        22

    accuracy                           0.93        60
   macro avg       0.94      0.94      0.94        60
weighted avg       0.94      0.93      0.93        60
```



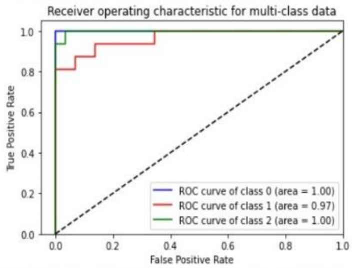

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the opti
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the opti
  % self.max_iter, ConvergenceWarning)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optimi
  % self.max_iter, ConvergenceWarning)
```




```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
```

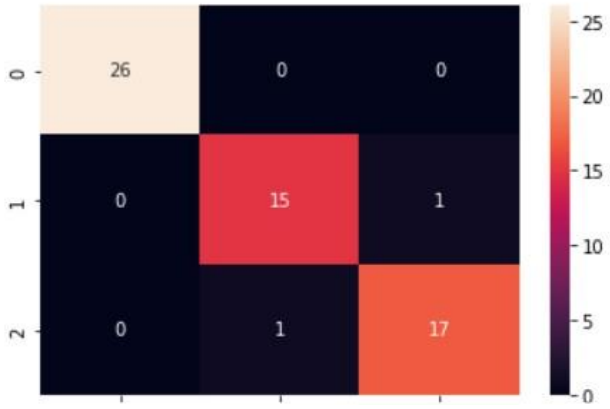
```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
```



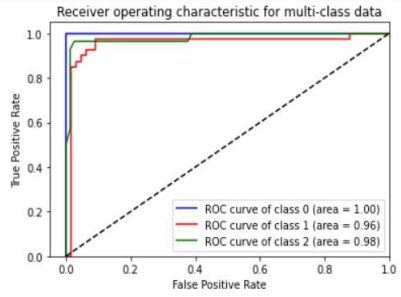

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optir
  % self.max_iter, ConvergenceWarning)
```

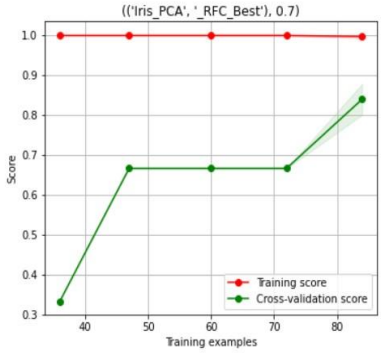
Random Forest classifier (without tuning)

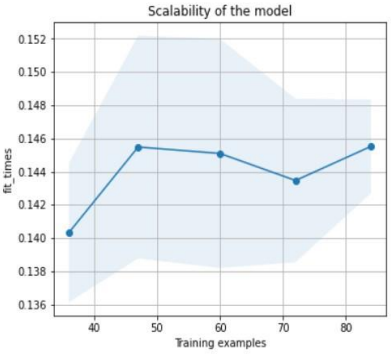
```
---------- (('Iris_PCA', '_RFC', {}), 0.39999999999999997) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        26
  versicolor       0.94      0.94      0.94        16
   virginica       0.94      0.94      0.94        18

    accuracy                           0.97        60
   macro avg       0.96      0.96      0.96        60
weighted avg       0.97      0.97      0.97        60
```

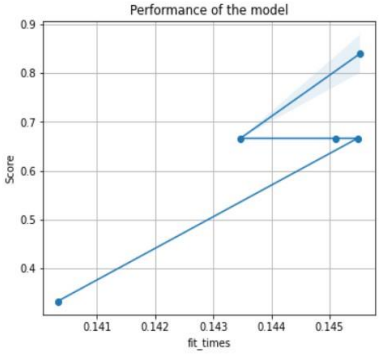


Random Forest classifier (with hyperparamter tuning)

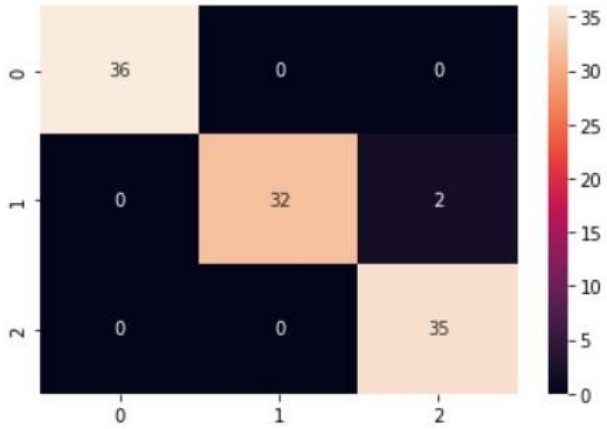









```
---------- (('Iris_PCA', '_RFC_Best'), 0.7) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=4, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        36
  versicolor       1.00      0.94      0.97        34
   virginica       0.95      1.00      0.97        35

    accuracy                           0.98       105
   macro avg       0.98      0.98      0.98       105
weighted avg       0.98      0.98      0.98       105
```



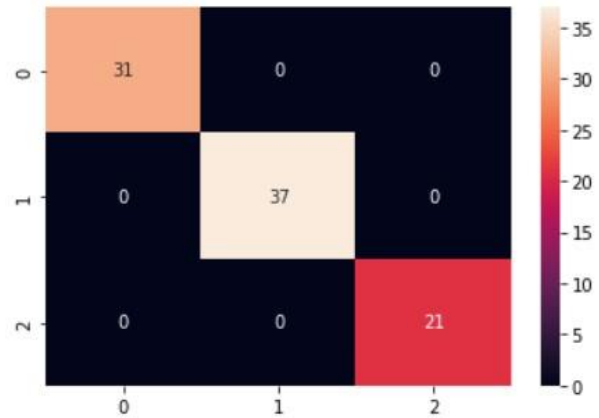
WINE dataset with PCA:-

```
scaler = MinMaxScaler()
X=scaler.fit_transform(X)

pca = PCA()
X_new = pca.fit_transform(X)
flRoutine(X_new, y, names, "Wine_PCA")
```

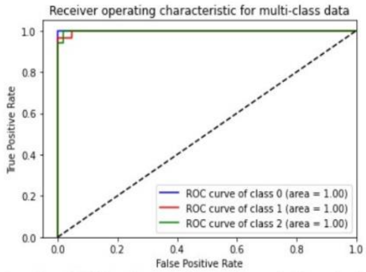
Output:-

SVM (without tuning)

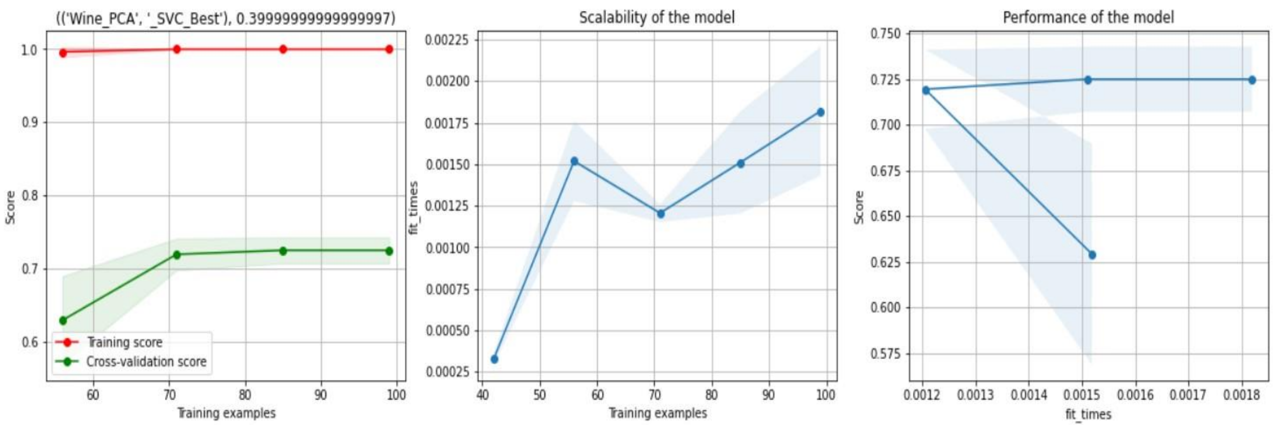
```
---------- (('Wine_PCA', '_SVC', {'kernel': 'linear'}), 0.5) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        31
     class_1       1.00      1.00      1.00        37
     class_2       1.00      1.00      1.00        21

    accuracy                           1.00        89
   macro avg       1.00      1.00      1.00        89
weighted avg       1.00      1.00      1.00        89
```



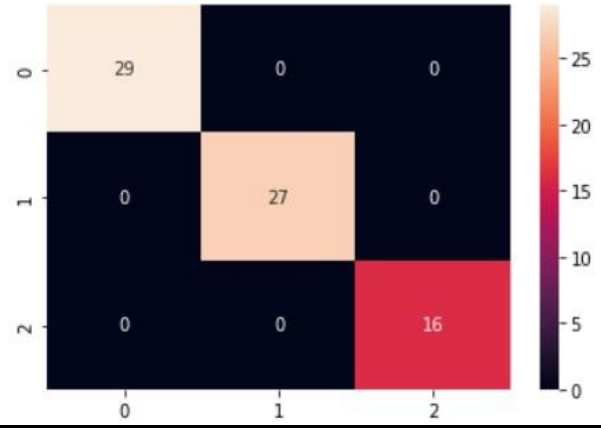
SVM (with hyperparamter tuning)




```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parame
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parame
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parame
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parame
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:536: FitFailedWarning: Estimator fit failed. The score on this train-test partition for these parame
ValueError: The number of classes has to be greater than one; got 1 class

  FitFailedWarning)
```

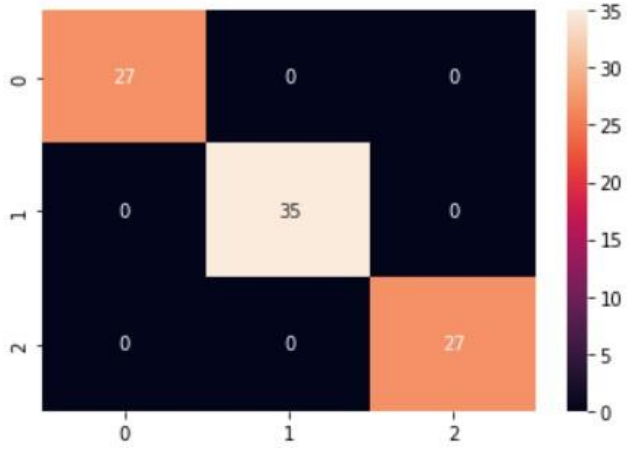


```
---------- (('Wine_PCA', '_SVC_Best'), 0.39999999999999997) ---------
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf', max_iter=-1,
    probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        29
     class_1       1.00      1.00      1.00        27
     class_2       1.00      1.00      1.00        16

    accuracy                           1.00        72
   macro avg       1.00      1.00      1.00        72
weighted avg       1.00      1.00      1.00        72
```

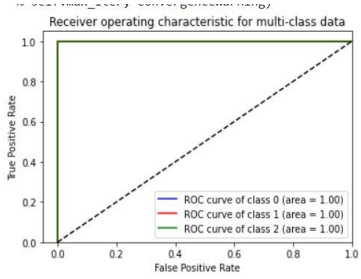


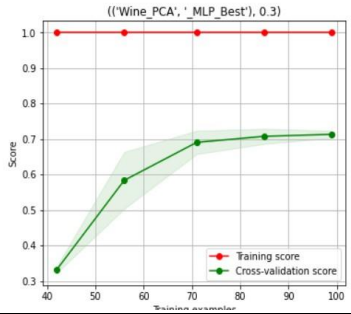
MLP (without tuning)

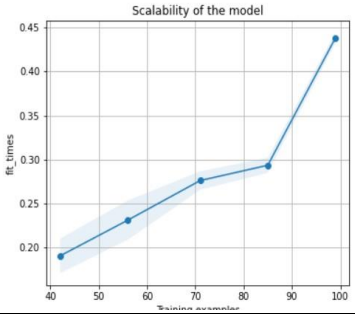
```
---------- (('Wine_PCA', '_MLP', {}), 0.5) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        27
     class_1       1.00      1.00      1.00        35
     class_2       1.00      1.00      1.00        27

    accuracy                           1.00        89
   macro avg       1.00      1.00      1.00        89
weighted avg       1.00      1.00      1.00        89
```

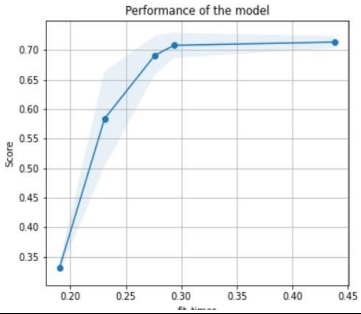


MLP (with hyperparamter tuning)

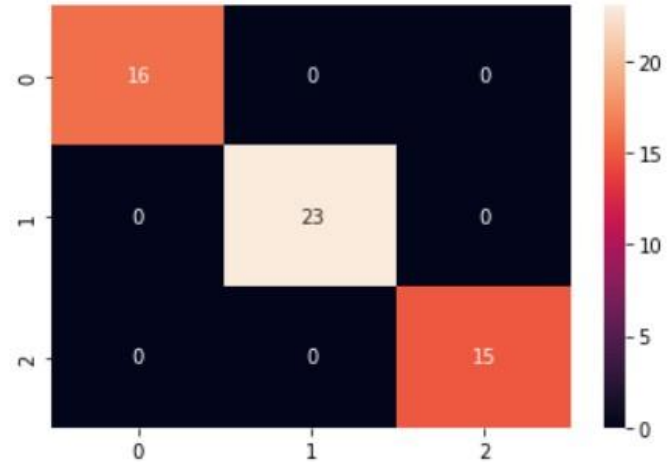






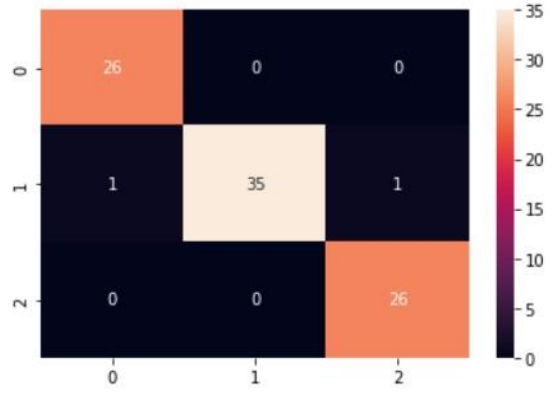



```
---------- (('Wine_PCA', '_MLP_Best'), 0.3) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        16
     class_1       1.00      1.00      1.00        23
     class_2       1.00      1.00      1.00        15

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```



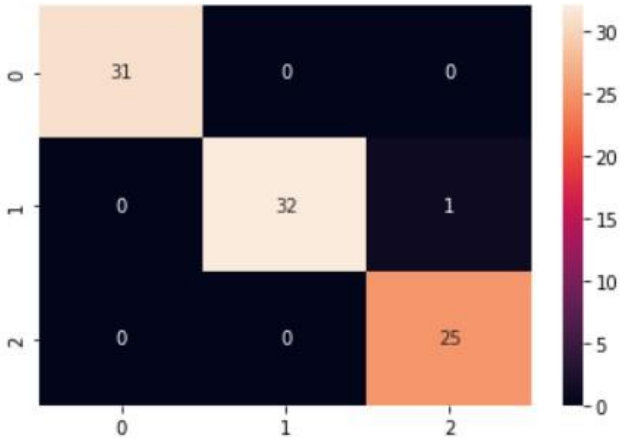
Random Forest Classifier (without tuning)

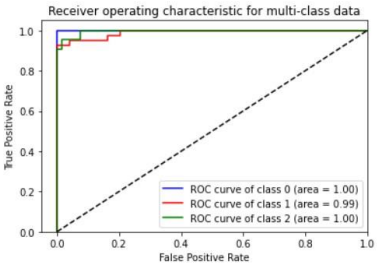
```
---------- (('Wine_PCA', '_RFC', {}), 0.5) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

     class_0       0.96      1.00      0.98        26
     class_1       1.00      0.95      0.97        37
     class_2       0.96      1.00      0.98        26

    accuracy                           0.98        89
   macro avg       0.98      0.98      0.98        89
weighted avg       0.98      0.98      0.98        89
```

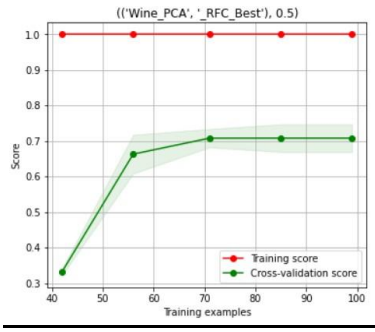


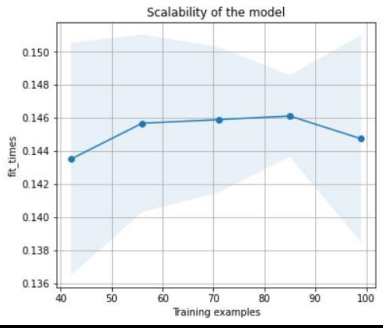
Random Forest Classifier (with hyperparameter tuning)

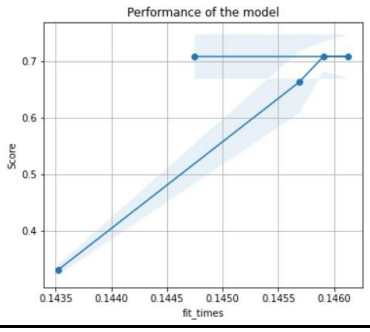
```
---------- (('Wine_PCA', '_RFC_Best'), 0.5) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=6, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

     class_0       1.00      1.00      1.00        31
     class_1       1.00      0.97      0.98        33
     class_2       0.96      1.00      0.98        25

    accuracy                           0.99        89
   macro avg       0.99      0.99      0.99        89
weighted avg       0.99      0.99      0.99        89
```











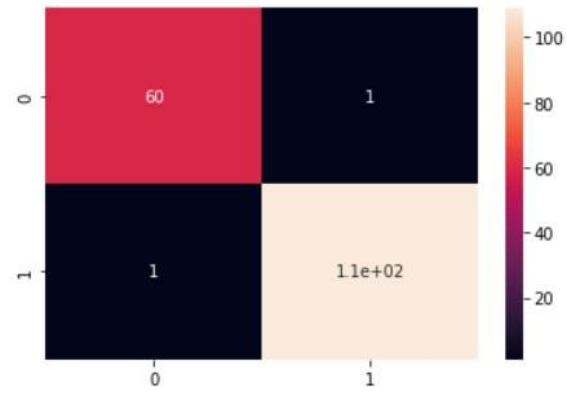
WBC dataset with PCA:-

```
scaler = MinMaxScaler()
X=scaler.fit_transform(X)

pca = PCA()
X_new = pca.fit_transform(X)
flRoutine(X_new, y, names, "WBC_PCA")
```

Output:-

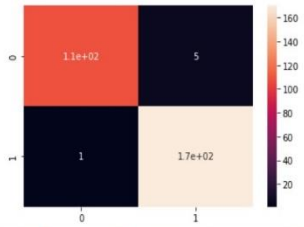
SVM (without tuning)

```
---------- (('WBC_PCA', '_SVC', {'kernel': 'rbf'}), 0.3) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

   malignant       0.98      0.98      0.98        61
      benign       0.99      0.99      0.99       110

    accuracy                           0.99       171
   macro avg       0.99      0.99      0.99       171
weighted avg       0.99      0.99      0.99       171
```



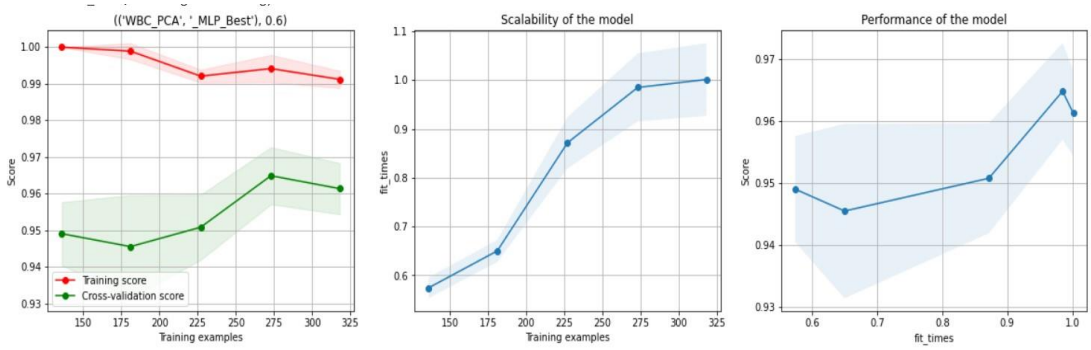
MLP (without tuning)

```
---------- (('WBC_PCA', '_MLP', {}), 0.5) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=200,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

   malignant       0.99      0.96      0.97       114
      benign       0.97      0.99      0.98       171

    accuracy                           0.98       285
   macro avg       0.98      0.98      0.98       285
weighted avg       0.98      0.98      0.98       285
```

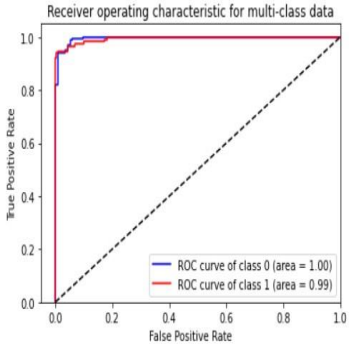



```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimi
  % self.max_iter, ConvergenceWarning)
```

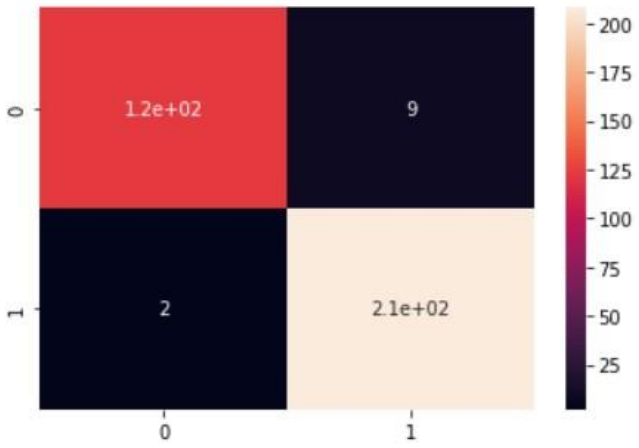
MLP (with hyperparameter tuning)



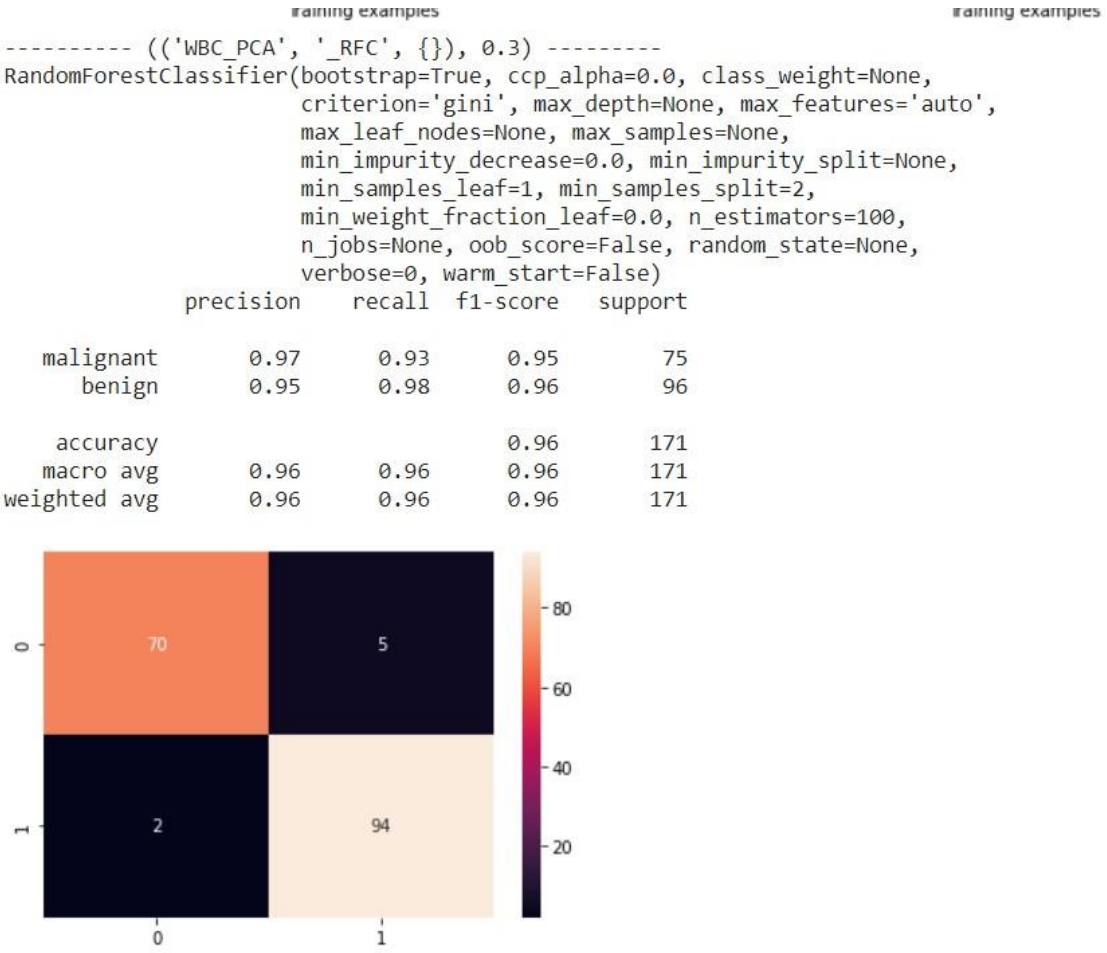

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:571: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (500) reached and the optii
  % self.max_iter, ConvergenceWarning)
```



```
---------- (('WBC_PCA', '_MLP_Best'), 0.6) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.6, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

   malignant       0.98      0.93      0.96       132
      benign       0.96      0.99      0.97       210

    accuracy                           0.97       342
   macro avg       0.97      0.96      0.97       342
weighted avg       0.97      0.97      0.97       342
```

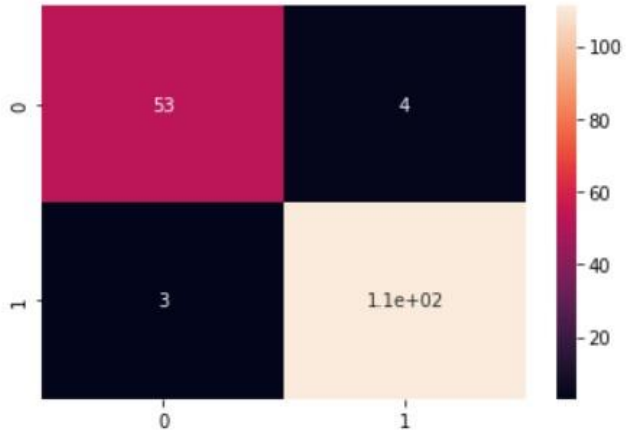


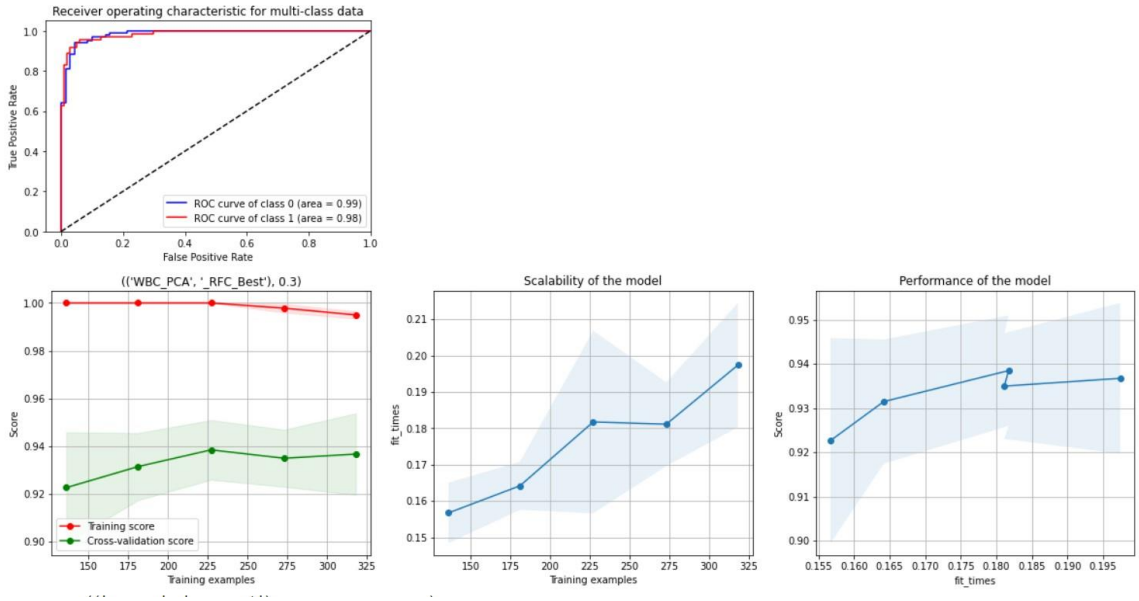
Random Forest Classifier (without tuning)



Random Forest Classifier (with hyperparamter tuning)

```
---------- (('WBC_PCA', '_RFC_Best'), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=6, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

   malignant       0.95      0.93      0.94        57
      benign       0.97      0.97      0.97       114

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.95       171
weighted avg       0.96      0.96      0.96       171
```





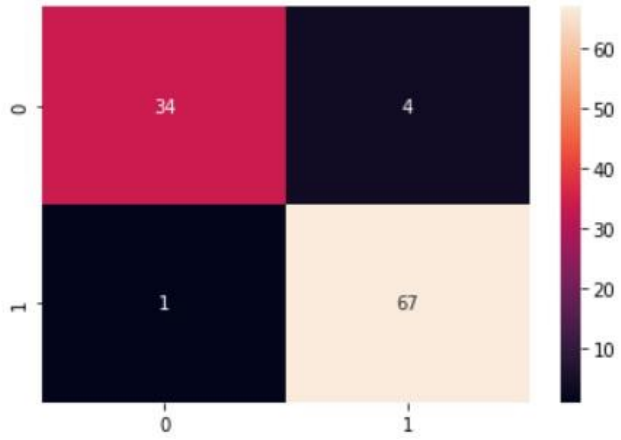
Ionosphere dataset with PCA:-

```
scaler = MinMaxScaler()
X=scaler.fit_transform(X)

pca = PCA()
X_new = pca.fit_transform(X)
flRoutine(X_new, y, names, "Ionosphere_PCA")
```

Output:-

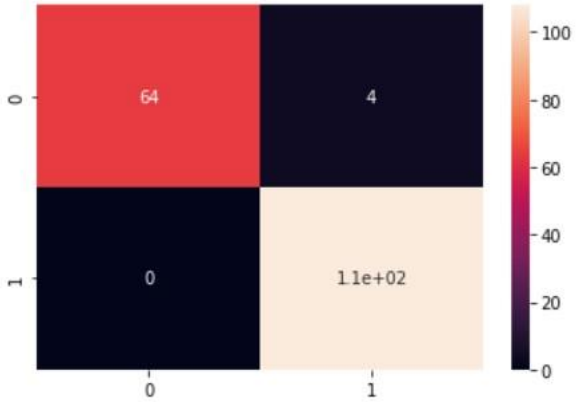
SVM (without tuning)

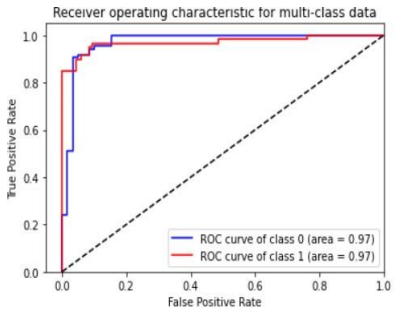
```
---------- (('Ionosphere_PCA', '_SVC', {'kernel': 'rbf'}), 0.3) ---------
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

           g       0.97      0.89      0.93        38
           b       0.94      0.99      0.96        68

    accuracy                           0.95       106
   macro avg       0.96      0.94      0.95       106
weighted avg       0.95      0.95      0.95       106
```

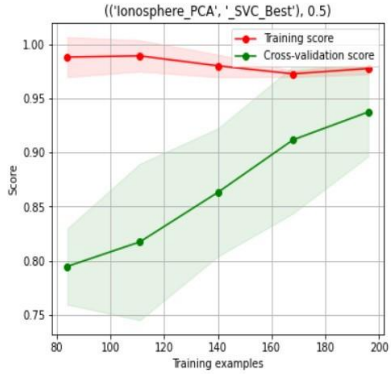


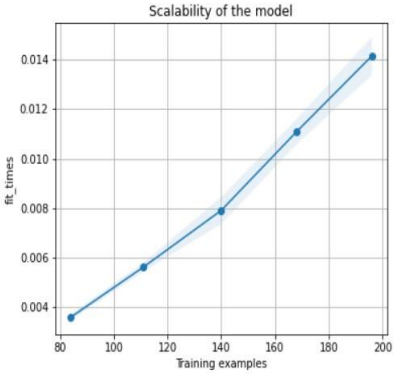
SVM (with hyperparameter tuning)

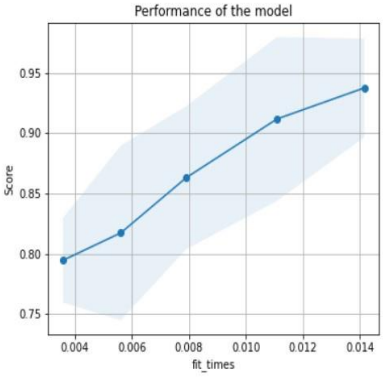
```
---------- (('Ionosphere_PCA', '_SVC_Best'), 0.5) ---------
SVC(C=1, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf', max_iter=-1,
    probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
              precision    recall  f1-score   support

           g       1.00      0.94      0.97        68
           b       0.96      1.00      0.98       108

    accuracy                           0.98       176
   macro avg       0.98      0.97      0.98       176
weighted avg       0.98      0.98      0.98       176
```





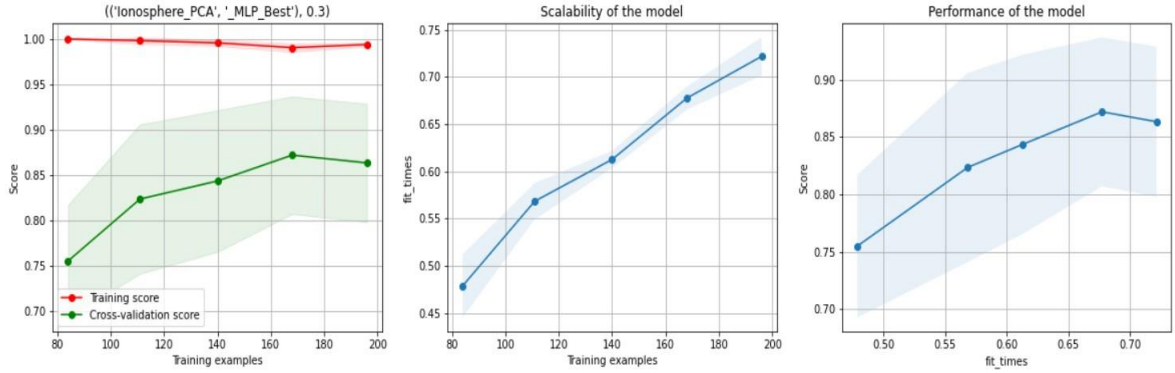


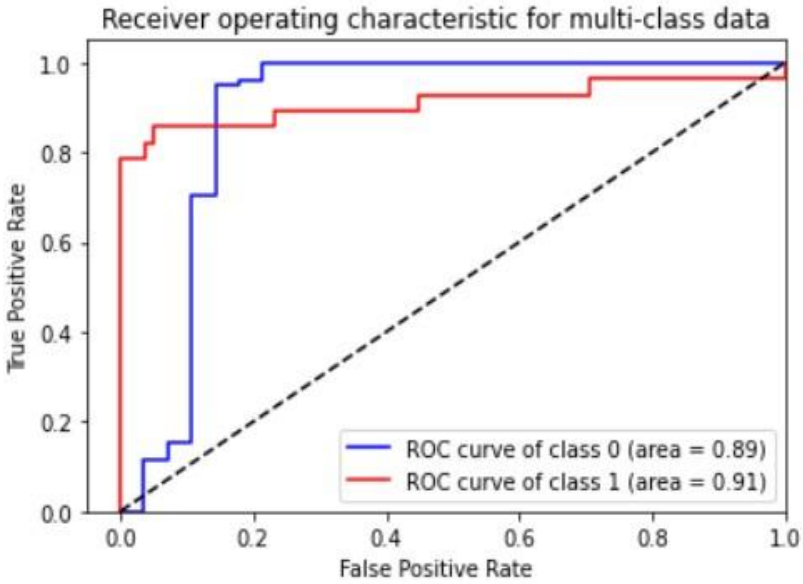




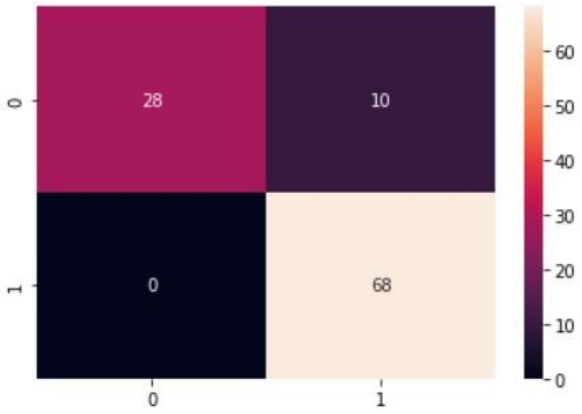
MLP (without tuning)

MLP (with hyperparameter tuning)



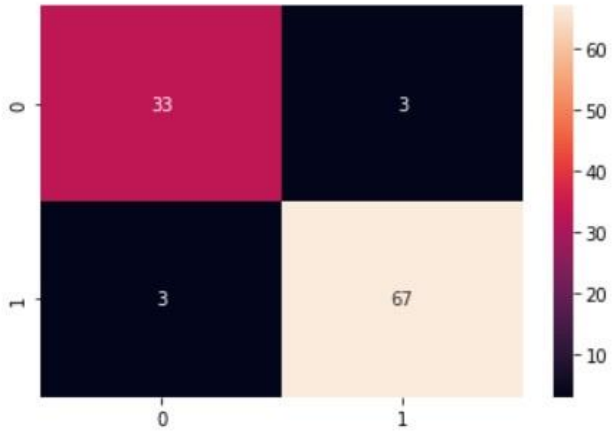


```
---------- (('Ionosphere_PCA', '_MLP_Best'), 0.3) ---------
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(100,), learning_rate='adaptive',
              learning_rate_init=0.001, max_fun=15000, max_iter=500,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=None, shuffle=True, solver='adam',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
              precision    recall  f1-score   support

           g       1.00      0.74      0.85        38
           b       0.87      1.00      0.93        68

    accuracy                           0.91       106
   macro avg       0.94      0.87      0.89       106
weighted avg       0.92      0.91      0.90       106
```



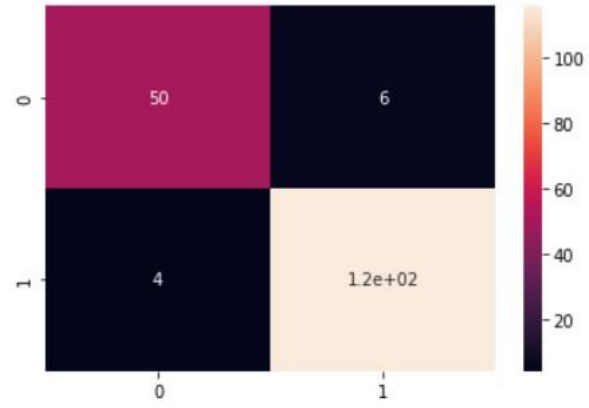
Random Forest Classifier (without tuning)

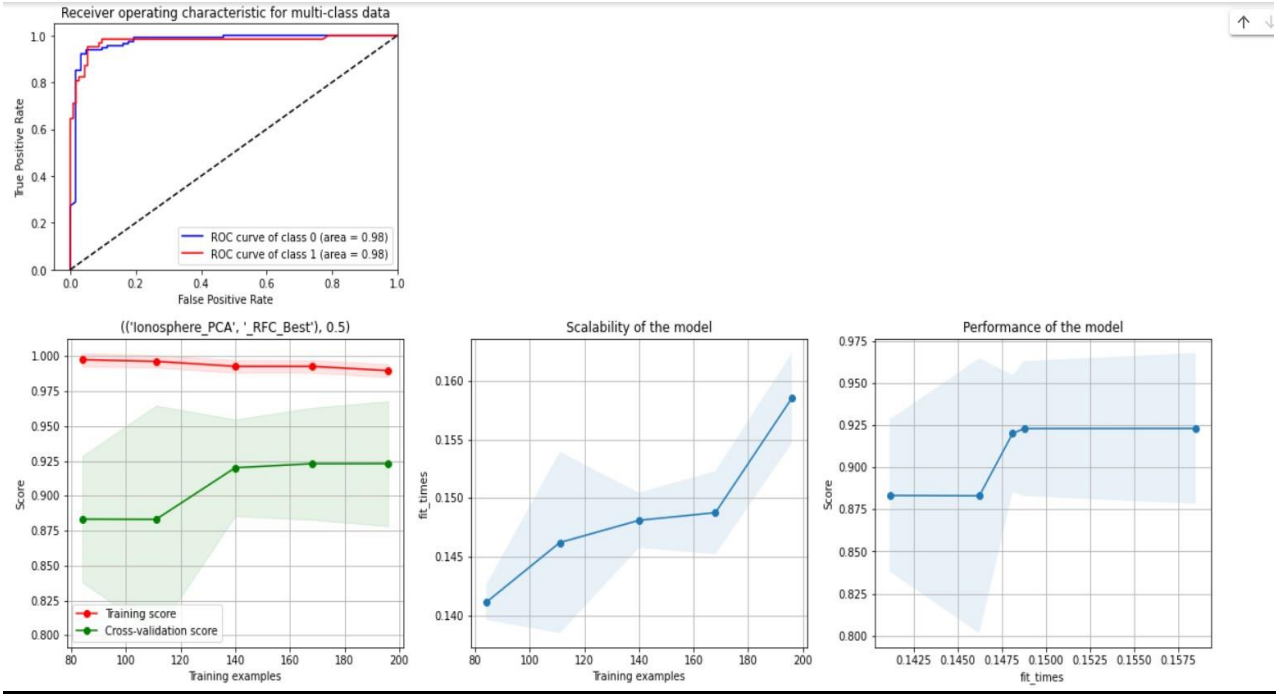
```
---------- (('Ionosphere_PCA', '_RFC', {}), 0.3) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

           g       0.92      0.92      0.92        36
           b       0.96      0.96      0.96        70

    accuracy                           0.94       106
   macro avg       0.94      0.94      0.94       106
weighted avg       0.94      0.94      0.94       106
```



Random Forest Classifier (with hyperparamter tuning)

```
---------- (('Ionosphere_PCA', '_RFC_Best'), 0.5) ---------
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=4, max_features='log2',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
              precision    recall  f1-score   support

           g       0.93      0.89      0.91        56
           b       0.95      0.97      0.96       120

    accuracy                           0.94       176
   macro avg       0.94      0.93      0.93       176
weighted avg       0.94      0.94      0.94       176
```





Dataset	Classifier	Train : Test Split	Accuracy
Iris	SVM (without tuning)	70:30	1.00
		60:40	0.98
		50:50	0.99
		40:60	0.98
		30:70	0.98
	SVM (with tuning)	70:30	0.96
		60:40	0.95
		50:50	0.96
		40:60	0.96
		30:70	0.96
	MLP (without tuning)	70:30	0.96
		60:40	0.98
		50:50	0.97
		40:60	0.98
		30:70	0.87
	MLP (with tuning)	70:30	0.98
		60:40	0.97

		50:50	0.99
		40:60	0.98
		30:70	0.95
	Random Forest (without tuning)	70:30	0.96
		60:40	0.95
		50:50	0.97
		40:60	0.91
		30:70	0.95
	Random Forest (with tuning)	70:30	0.98
		60:40	0.97
		50:50	0.96
		40:60	0.96
		30:70	0.95
Wine	SVM (without tuning)	70:30	0.98
		60:40	0.94
		50:50	0.98
		40:60	0.93
		30:70	0.89
	SVM (with tuning)	70:30	0.98
		60:40	0.99
		50:50	0.96
		40:60	0.92
		30:70	0.83
	MLP (without tuning)	70:30	0.46
		60:40	0.57
		50:50	0.91
		40:60	0.50
		30:70	0.93
	MLP (with tuning)	70:30	0.50
		60:40	0.97
		50:50	0.96
		40:60	0.61
		30:70	0.36
	Random Forest (without tuning)	70:30	0.98
		60:40	0.97
		50:50	0.91
		40:60	0.97
		30:70	0.94
		70:30	1.00

	Random Forest (with tuning)	60:40	0.99
		50:50	0.98
		40:60	0.97
		30:70	0.98
Women Breast Cancer	SVM (without tuning)	70:30	0.95
		60:40	0.96
		50:50	0.96
		40:60	0.94
		30:70	0.95
	MLP (without tuning)	70:30	0.96
		60:40	0.92
		50:50	0.94
		40:60	0.91
		30:70	0.85
	MLP (with tuning)	70:30	0.92
		60:40	0.93
		50:50	0.94
		40:60	0.89
		30:70	0.93
	Random Forest (without tuning)	70:30	0.94
		60:40	0.95
		50:50	0.97
		40:60	0.94
		30:70	0.96
	Random Forest (with tuning)	70:30	0.97
		60:40	0.97
		50:50	0.96
		40:60	0.93
		30:70	0.95
Ionosphere	SVM (without tuning)	70:30	0.97
		60:40	0.92
		50:50	0.95
		40:60	0.95
		30:70	0.95
	SVM (with tuning)	70:30	0.87
		60:40	0.87
		50:50	0.89
		40:60	0.93
		30:70	0.92

	MLP(without tuning)	70:30	0.92
		60:40	0.92
		50:50	0.89
		40:60	0.85
		30:70	0.86
	MLP (with tuning)	70:30	0.93
		60:40	0.91
		50:50	0.88
		40:60	0.87
		30:70	0.87
	Random Forest (without tuning)	70:30	0.96
		60:40	0.96
		50:50	0.92
		40:60	0.93
		30:70	0.91
	Random Forest (with tuning)	70:30	0.88
		60:40	0.92
		50:50	0.91
		40:60	0.92
		30:70	0.93

Dataset	Classifier	Accuracy	Precision	Recall	F1-score
Iris	SVM (without tuning)	**1.00**	1.00	1.00	1.00
			1.00	1.00	1.00
			1.00	1.00	1.00
	SVM (with tuning)	0.96	1.00	1.00	1.00
			0.93	0.93	0.93
			0.93	0.93	0.93
	MLP (without tuning)	0.98	1.00	1.00	1.00
			1.00	0.94	0.97
			0.91	1.00	0.97
	MLP (with tuning)	0.99	1.00	1.00	1.00
			0.95	1.00	0.97
			1.00	0.96	0.98
	Random Forest	0.97	1.00	1.00	1.00
			1.00	0.92	0.96

	(without tuning)		0.92	1.00	0.96
	Random Forest (with tuning	0.98	1.00	1.00	1.00
			1.00	0.94	0.97
			0.93	1.00	0.97
Wine	SVM (without tuning)				
	SVM (with tuning)				
	MLP (without tuning)				
	MLP (with tuning)	0.36	0.38	1.00	0.55
			0.00	0.00	0.00
			0.00	0.00	0.00
	Random Forest (without tuning)	0.98	0.95	1.00	0.97
			1.00	0.94	0.97
			1.00	1.00	1.00
	Random Forest (with tuning)	**1.00**	1.00	1.00	1.00
			1.00	1.00	1.00
			1.00	1.00	1.00
Wbc	SVM (without tuning)	0.96	0.96	0.94	0.95
			0.96	0.98	0.97
	MLP (without tuning)	0.94	0.96	0.98	0.93
			0.93	0.89	0.95
	MLP (with tuning)	0.93	0.99	0.84	0.91
			0.90	0.99	0.94
	Random Forest (without tuning)	**0.97**	0.98	0.93	0.95
			0.96	0.99	0.98
	Random Forest	**0.97**	0.98	0.94	0.96

	(with tuning		0.96	0.99	0.98
Ionosphere	SVM (without tuning)	**<u>0.97</u>**	1.00	0.93	0.96
			0.95	1.00	0.98
	SVM (with tuning)	0.93	0.89	0.93	0.91
			0.95	0.93	0.94
	MLP (without tuning)	0.92	0.94	0.82	0.88
			0.80	0.97	0.93
	MLP (with tuning)	0.93	0.97	0.85	0.91
			0.92	0.98	0.95
	Random Forest (without tuning)	0.96	0.97	0.91	0.94
			0.96	0.99	0.97
	Random Forest (with tuning	0.93	0.97	0.83	0.89
			0.90	0.99	0.94